THE DL_POLY_3 USER MANUAL

I.T. Todorov and W. Smith

CCLRC Daresbury Laboratory
Daresbury, Warrington WA4 4AD
Cheshire, UK

Version 3.06, March 2006

ABOUT DL_POLY_3

DL_POLY_3 is a parallel molecular dynamics simulation package developed at Daresbury Laboratory by W. Smith and I.T. Todorov. The package was developed under the auspices of the Engineering and Physical Sciences Research Council (EPSRC) for the EPSRC's Collaborative Computational Project for the Computer Simulation of Condensed Phases (CCP5), the Computational Chemistry Group at Daresbury Laboratory and the NERC e-Science Project *Computational Chemistry in the Environment*, directed by M. Dove.

DL_POLY_3 is the property of Daresbury Laboratory and is issued free **under licence** to academic institutions pursuing scientific research of a non-commercial nature. A handling fee is charged for new DL_POLY_3 licence for NON-UK users. Commercial organisations may be permitted a licence to use the package after negotiation with the owners. Daresbury Laboratory is the sole centre for distribution of the package. Under no account is it to be redistributed to third parties without consent of the owners.

The purpose of the DL_POLY_3 package is to provide software for academic research that is inexpensive, accessible and free of commercial considerations. Users have direct access to source code for modification and inspection. In the spirit of the enterprise, contributions in the form of working code are welcome, provided the code is compatible with DL_POLY_3 in regard to its interfaces and programming style and it is adequately documented.

DISCLAIMER

Neither the CCLRC, EPSRC, NERC, CCP5 nor any of the authors of the DL_POLY_3 package or its derivatives guarantee that the package is free from error. Neither do they accept responsibility for any loss or damage that results from its use.

ACKNOWLEDGEMENTS

DL_POLY_3 was developed at the Central Laboratory of the Research Councils, Daresbury Laboratory, with support from the Engineering and Physical Sciences Research Council and the Natural Environment Research Council. The package is the property of the Council for the Central Laboratory of the Research Councils of the United Kingdom.

Advice, assistance and encouragement in the development of DL_POLY_3 has been given by many people. We gratefully acknowledge the following:

T.R. Forester, Ian Bush, M.F. Guest, M. Leslie, R.J. Allan, D. Tildesley, M. Pinches, D. Rapaport, the UK's Materials Consortium under C.R.A. Catlow and the NERC e-Science project "Computational Chemistry in the Environment" under M. Dove.

Manual Notation

In the DL_POLY manuals specific fonts are used to convey specific meanings:

1. *directories* - indicates unix file directories
2. ROUTINES - indicates subroutines, functions and programs.
3. *macros* - indicates a macro (file of unix commands)
4. **directive** - indicates directives or keywords
5. `variables` - indicates named variables and parameters
6. FILE - indicates filenames.

Contents

Title Page a

About DL_POLY_3 . . . i
Disclaimer . . . ii
Acknowledgements . . . iii
Manual Notation . . . iv

Contents v

List of Tables x

List of Figures xi

1 Introduction 1

1.1 The DL_POLY Package . . . 2
1.2 Functionality . . . 2
1.2.1 Molecular Systems . . . 2
1.2.2 Force Field . . . 3
1.2.3 Boundary Conditions . . . 3
1.2.4 Java Graphical User Interface . . . 4
1.2.5 Algorithms . . . 4
1.3 Programming Style . . . 4
1.3.1 Programming Language . . . 5
1.3.2 Modularisation and Intent . . . 5
1.3.3 Memory Management . . . 5
1.3.4 Target Computers . . . 5
1.3.5 Version Control System (CVS) . . . 5
1.3.6 Internal Documentation . . . 6
1.3.7 FORTRAN 90 Parameters and Arithmetic Precision . . . 6
1.3.8 Units . . . 6
1.3.9 Error Messages . . . 7

1.4 The DL_POLY_3 Directory Structure . . . 7
1.4.1 The *source* Sub-directory . . . 8
1.4.2 The *utility* Sub-directory . . . 8
1.4.3 The *data* Sub-directory . . . 8
1.4.4 The *bench* Sub-directory . . . 8
1.4.5 The *execute* Sub-directory . . . 8
1.4.6 The *build* Sub-directory . . . 8
1.4.7 The *public* Sub-directory . . . 8
1.4.8 The *java* Sub-directory . . . 9
1.5 Obtaining the Source Code . . . 9
1.6 Other Information . . . 10

2 Force Fields **11**
2.1 Introduction to DL_POLY_3 Force Field . . . 12
2.2 The Intramolecular Potential Functions . . . 13
2.2.1 Bond Potentials . . . 13
2.2.2 Distance Restraints . . . 15
2.2.3 Valence Angle Potentials . . . 15
2.2.4 Angular Restraints . . . 17
2.2.5 Dihedral Angle Potentials . . . 18
2.2.6 Improper Dihedral Angle Potentials . . . 20
2.2.7 Inversion Angle Potentials . . . 21
2.2.8 Tethering Forces . . . 23
2.2.9 Frozen Atoms . . . 24
2.3 The Intermolecular Potential Functions . . . 24
2.3.1 Short Ranged (van der Waals) Potentials . . . 25
2.3.2 Metal Potentials . . . 27
2.3.3 Tersoff Potential . . . 28
2.3.4 Three Body Potentials . . . 31
2.3.5 Four Body Potentials . . . 32
2.4 Long Ranged Electrostatic (coulombic) Potentials . . . 33
2.4.1 Direct Coulomb Sum . . . 33
2.4.2 Force Shifted Coulomb Sum . . . 34
2.4.3 Coulomb Sum with Distance Dependent Dielectric . . . 35
2.4.4 Reaction Field . . . 35
2.4.5 Smoothed Particle Mesh Ewald . . . 36
2.5 Polarisation Shell Models . . . 39

2.5.1 Dynamical (Adiabatic Shells) . . . 40
2.5.2 Relaxed (Massless Shells) . . . 40
2.6 External Fields . . . 41

3 Integration Algorithms 42
3.1 Introduction . . . 43
3.2 Bond Constraints . . . 45
3.3 Potential of Mean Force (PMF) Constraints and the Evaluation of Free Energy . . . 47
3.4 Thermostats . . . 48
3.4.1 Langevin Thermostat . . . 48
3.4.2 Evans Thermostat (Gaussian Constraints) . . . 50
3.4.3 Berendsen Thermostat . . . 51
3.4.4 Nosé-Hoover Thermostat . . . 52
3.5 Barostats . . . 54
3.5.1 Instantaneous pressure and stress . . . 55
3.5.2 Berendsen Barostat . . . 55
3.5.3 The Hoover Barostat . . . 56
3.5.4 Martyna-Tuckerman-Klein Barostat . . . 60

4 Construction and Execution 61
4.1 Constructing DL_POLY_3 : an Overview . . . 62
4.1.1 Constructing the Standard Version . . . 62
4.1.2 Constructing Non-standard Versions . . . 63
4.2 Compiling and Running DL_POLY_3 . . . 64
4.2.1 Compiling the DL_POLY_3 Source Code . . . 64
4.2.2 Running DL_POLY_3 . . . 66
4.2.3 Restarting DL_POLY_3 . . . 67
4.2.4 Notes on the DL_POLY_3 Simulation Efficiency and Performance . . . 67
4.3 A Guide to Preparing Input Files . . . 68
4.3.1 Inorganic Materials . . . 69
4.3.2 Macromolecules . . . 69
4.3.3 Adding Solvent to a Structure . . . 70
4.3.4 Analysing Results . . . 70
4.3.5 Choosing Ewald Sum Variables . . . 70
4.4 DL_POLY_3 Warning and Error Processing . . . 72
4.4.1 The DL_POLY_3 Internal Warning Facility . . . 72
4.4.2 The DL_POLY_3 Internal Error Facility . . . 72

5 Data Files **73**
5.1 The INPUT Files . . . 74
5.1.1 The CONTROL File . . . 74
5.1.2 The CONFIG File . . . 83
5.1.3 The FIELD File . . . 86
5.1.4 The REFERENCE File . . . 100
5.1.5 The REVOLD File . . . 101
5.1.6 The TABLE File . . . 102
5.2 The OUTPUT Files . . . 103
5.2.1 The HISTORY File . . . 104
5.2.2 The DEFECTS File . . . 105
5.2.3 The OUTPUT File . . . 107
5.2.4 The REVCON File . . . 110
5.2.5 The REVIVE File . . . 110
5.2.6 The RDFDAT File . . . 110
5.2.7 The ZDNDAT File . . . 111
5.2.8 The STATIS File . . . 111

6 DL_POLY_3 Parallelisation, Source Code Modularisation and Structure **114**
6.1 DL_POLY_3 Parallelisation . . . 115
6.1.1 The Domain Decomposition Strategy . . . 115
6.1.2 Distributing the Intramolecular Bonded Terms . . . 116
6.1.3 Distributing the Non-bonded Terms . . . 117
6.1.4 Modifications for the Ewald Sum . . . 118
6.1.5 Metal Potentials . . . 118
6.1.6 Tersoff, Three Body and Four Body Potentials . . . 119
6.1.7 Globally Summed Properties . . . 119
6.1.8 The Parallel (DD tailored) SHAKE and RATTLE Algorithms . . . 119
6.2 DL_POLY_3 Source Code . . . 120
6.2.1 DL_POLY_3 Modularisation Principles . . . 120
6.2.2 DL_POLY_3 File Structure . . . 121
6.2.3 Module Files . . . 123
6.2.4 General Files . . . 123
6.2.5 VV and LFV Specific Files . . . 123
6.2.6 SERIAL Specific Files . . . 123
6.2.7 Comments on MPI Handling . . . 124
6.2.8 Comments on SETUP_MODULE . . . 124

7 Examples **126**
7.1 Test Cases . . . 127
7.1.1 Test Case 1 and 2: Sodium Chloride . . . 127
7.1.2 Test Case 3 and 4: DPMC in Water . . . 127
7.1.3 Test Case 5 and 6: $KNaSi_2O_5$ . . . 127
7.1.4 Test Case 7 and 8: Gramicidin A molecules in Water . . . 127
7.1.5 Test Case 9 and 10: SiC with Tersoff Potentials . . . 127
7.1.6 Test Case 11 and 12: Cu_3Au with Metal Potentials . . . 128
7.1.7 Test Case 13 and 14: lipid bilayer in water . . . 128
7.1.8 Test Case 15 and 16: relaxed and adiabatic shell model MgO . . . 128
7.1.9 Test Case 17 and 18: Potential of mean force on K+ in water MgO . . . 128
7.2 Benchmark Cases . . . 128

Appendices **129**

A DL_POLY_3 Periodic Boundary Conditions **129**

B DL_POLY_3 Macros **132**

C DL_POLY_3 Makefiles **136**

D DL_POLY_3 Error Messages and User Action **154**

E DL_POLY_3 README **192**

Bibliography **195**

Index **198**

List of Tables

5.1 Internal Trajectory/Defects File Key 79
5.2 Internal Restart Key 80
5.3 Internal Ensemble Key 80
5.4 Electrostatics Key 83
5.5 CONFIG File Key (record 2) 85
5.6 Periodic Boundary Key (record2) 85
5.7 Tethering Potentials 91
5.8 Chemical Bond Potentials 92
5.9 Valence Angle Potentials 93
5.10 Dihedral Angle Potentials 94
5.11 Inversion Angle Potentials 95
5.12 Pair Potentials 96
5.13 Metal Potential 97
5.14 Tersoff Potential 98
5.15 Three-body Potentials 99
5.16 Four-body Potentials 100
5.17 External Fields 101

List of Figures

5.1 DL_POLY_3 input (left) and output (right) files. Note: files marked with an asterisk are non-mandatory. 74

Chapter 1

Introduction

Scope of Chapter

This chapter describes the concept, design and directory structure of DL_POLY_3 and how to obtain a copy of the source code.

1.1 The DL_POLY Package

DL_POLY [1] is a package of subroutines, programs and data files, designed to facilitate molecular dynamics simulations of macromolecules, polymers, ionic systems and solutions on a distributed memory parallel computer. It is available in two forms: DL_POLY_2 (written by Bill Smith and Tim Forester) and DL_POLY_3 (written by Bill Smith and Ilian Todorov) [2]. Both versions were originally written on behalf of CCP5, the UK's Collaborative Computational Project on Molecular Simulation, which has been in existence since 1980 ([3], http://www.ccp5.ac.uk). Further developments of DL_POLY_3 were fostered by the NERC eScience Project on *Computational Chemistry in the Environment* headed by Martin Dove at Cambridge.

The two forms of DL_POLY differ primarily in their method of exploiting parallelism. DL_POLY_2 uses a Replicated Data (RD) strategy [4, 5, 6, 7] which works well for computing systems of order 100 processors and molecular simulations up to order 30,000 atoms. DL_POLY_3 is based on the Domain Decomposition (DD) strategy [2, 8, 9, 4, 5], and is best suited to large computer systems (to any number of processors presentable as 2_N), and molecular simulations from 10^3 to 10^9 atoms. The two packages are reasonably compatible, so that it is possible to scale up from a DL_POLY_2 to a DL_POLY_3 simulation with very little effort. It should be apparent from these comments that DL_POLY_3 is not intended as a replacement for DL_POLY_2.

Users are reminded that we are interested in hearing what other features could be usefully incorporated. We obviously have ideas of our own and CCP5 strongly influences developments, but other input would be welcome nevertheless. We also request that our users respect the integrity of DL_POLY_3 source and not pass it on to third parties. We require that all users of the package register with us, not least because we need to keep everyone abreast of new developments and discovered bugs. We have developed various forms of licence, which we hope will ward off litigation (from both sides), without denying access to genuine scientific users.

Further information on the DL_POLY packages may be obtained from the project website:

$$http://www.cse.clrc.ac.uk/msi/software/DL_POLY \quad .$$

1.2 Functionality

The following is a list of the features DL_POLY_3 supports.

1.2.1 Molecular Systems

DL_POLY_3 will simulate the following molecular species:

- Simple atomic systems and mixtures, e.g. Ne, Ar, Kr, etc.
- Simple unpolarisable point ions, e.g. NaCl, KCl, etc.
- Polarisable point ions and molecules, e.g. MgO, H_2O etc.
- Polymers with rigid bonds, e.g. C_nH_{2n+2}
- Polymers with rigid bonds and point charges, e.g. proteins, macromolecules etc.
- Silicate glasses and zeolites

- Simple metals and metal alloys, e.g. Al, Ni, Cu, Cu_3Au etc.
- Hydro-carbons and transition elements, e.g. C, Si, Ge, SiC, SiGe, ets.

Note that DL_POLY_3 does not handle molecules containing rigid bodies. This is a major distinction from DL_POLY_2.

1.2.2 Force Field

The DL_POLY_3 force field includes the following features:

1. All common forms of non-bonded atom-atom potentials
2. Atom-atom (site-site) coulombic potentials
3. Sutton-Chen (density dependent potentials) (for metals) [10, 11, 12]
4. Tersoff (density dependent potentials) (for hydro-carbons) [13]
5. Three-body valence angle and hydrogen bond potentials
6. Four-body inversion potentials
7. Ion core-shell polarasation
8. Tether potentials
9. Chemical bond potentials
10. Valence angle potentials
11. Dihedral angle (and improper dihedral angle) potentials
12. Inversion angle potentials
13. External fieldpotential!external field potentials.

The parameters describing these potentials may be obtained, for example, from the GROMOS [14], Dreiding [15] or AMBER [16] forcefield, which share functional forms. It is relatively easy to adapt DL_POLY_3 to user specific force fields.

1.2.3 Boundary Conditions

DL_POLY_3 will accommodate the following boundary conditions:

1. None, e.g. isolated molecules *in vacuo*
2. Cubic periodic boundaries
3. Orthorhombic periodic boundaries
4. Parallelepiped periodic boundaries
5. Slab (x,y periodic, z non-periodic)

These are describe in detail in Appendix A. Note that periodic boundary conditions (PBC) 1 and 5 above require careful consideration to enable efficient load balancing on a parallel computer.

1.2.4 Java Graphical User Interface

The DL_POLY_3 Graphical User Interface (GUI) is the same as that for DL_POLY_2, which is written in the Java programming language from Sun Microsystems. A major advantage of this is the free availability of the Java programming environment from Sun, and also its portability across platforms. The compiled GUI may be run without recompiling on any Java supported machine. The GUI is an integral component of the DL_POLY suite and is available on the same terms (see the GUI manual [17]).

1.2.5 Algorithms

1.2.5.1 Parallel Algorithms

DL_POLY_3 exclusively employs the Domain Decomposition parallelisation strategy [8, 9, 4, 5] (see Section 6.1.1).

1.2.5.2 Molecular Dynamics Algorithms

DL_POLY_3 offers a selection of MD integration algorithms couched in both Velocity Verlet (VV) and Leapfrog Verlet (LFV) manner [18]. These generate NVE, NVE$_{kin}$, NVT, NPT and N$\underline{\underline{\sigma}}$T ensembles with a selection of thermostats and barostats. Parallel versions of the RATTLE [19] and SHAKE [7] algorithms are used for solving bond constraints in the VV and LFV cast integrations respectively.

The following MD algorithms are available:

1. Constant E algorithm
2. Langevin constant T algorithm [20]
3. Evans constant E$_{kin}$ algorithm [21]
4. Berendsen constant T algorithm [22]
5. Hoover constant T algorithm [23]
6. Berendsen constant T,P algorithm [22]
7. Hoover constant T,P algorithm [23]
8. Martyna, Tuckerman and Klein (MTK) constant T,P algorithm [24]
9. Berendsen constant T,$\underline{\underline{\sigma}}$ algorithm [22]
10. Hoover constant T,$\underline{\underline{\sigma}}$ algorithm [23]
11. Martyna, Tuckerman and Klein (MTK) constant T,$\underline{\underline{\sigma}}$ algorithm [24].

1.3 Programming Style

The programming style of DL_POLY_3 is intended to be as uniform as possible. The following stylistic rules apply throughout. Potential contributors of code are requested to note the stylistic convention.

1.3.1 Programming Language

DL_POLY_3 is written in free format FORTRAN 90. In DL_POLY_3 we have adopted the convention of *explicit type declaration* i.e. we have used

```
Implicit None
```

in all subroutines. Thus all variables must be given an explicit type: `Integer, Real(wp)` etc.

1.3.2 Modularisation and Intent

DL_POLY_3 exploits the full potential of the modularisation concept in FORTRAN 90. Variables having in common description of certain feature or method in DL_POLY_3 are grouped in modules. This simplifies subroutines' calling sequences and decreases error-proneness in programming as subroutines must define what they use and from which module. To decrease error-proneness further, arguments that are passed in calling sequences of functions or subroutines have defined intent, i.e. whether they are to be:

- passed in only (`Intent (In)`) - the argument is not allowed to be changed by the routine
- passed out only (`Intent (Out)`) - the "coming in" value of the argument is unimportant
- passed in both directions in and out (`Intent (InOut)`) - the "coming in" value of the argument is important and the argument is allowed to be changed.

1.3.3 Memory Management

DL_POLY_3 exploits the dynamic array allocation features of FORTRAN 90 to assign the necessary array dimensions.

1.3.4 Target Computers

DL_POLY_3 is intended for distributed memory parallel computers. It was developed on Cray T3E and IBM SP4 architectures.

Compilation of DL_POLY_3 in parallel mode requires **only** a FORTRAN 90 compiler and Message Passing Interface (MPI) to handle communications. Compilation of DL_POLY_3 in serial mode is also possible requires **only** a FORTRAN 90 compiler.

1.3.5 Version Control System (CVS)

DL_POLY_3 was developed with the aid of the CVS version control system. We strongly recommend that users of DL_POLY_3 adopt this system for local development of the code, particularly where several users access the same source code. For information on CVS please contact:

$$info - cvs - request@gnu.org$$

or visit the web site:

$$http://www.cyclic.com/ \quad .$$

1.3.6 Internal Documentation

All subroutines are supplied with a header block of FORTRAN 90 comment (!) records giving:

1. A CVS revision number and associated data
2. The name of the author and/or modifying author
3. The version number or date of production
4. A brief description of the function of the subroutine
5. A copyright statement.

Elsewhere FORTRAN 90 comment cards (!) are used liberally.

1.3.7 FORTRAN 90 Parameters and Arithmetic Precision

All global parameters defined by the FORTRAN 90 parameter statements are specified in the module file: SETUP_MODULE, which is included at compilation time in all subroutines requiring the parameters. All parameters specified in SETUP_MODULE are described by one or more comment cards.

One super-global parameter is defined at compilation time in the KINDS_F90 module file specifying the working precision (`wp`) by kind for real and complex variables and parameters. The default is 64-bit (double) precision, i.e. `Real(wp)`. Users wishing to compile the code with quadruple precision must ensure that their architecture and FORTRAN 90 compiler can allow that and then change the default in KINDS_F90. Changing the precision to anything else that is allowed by the FORTRAN 90 compiler and the machine architecture must also be compliant with the MPI working precision `mpi_wp` as defined in COMMS_MODULE (in such cases users must correct for that in there).

1.3.8 Units

Internally all DL_POLY_3 subroutines and functions assume the use of the following defined *molecular units*:

- The unit of time (t_o) is 1×10^{-12} seconds (i.e. picoseconds)
- The unit of length (ℓ_o) is 1×10^{-10} metres (i.e. Ångstroms)
- The unit of mass (m_o) is $1.6605402 \times 10^{-27}$ kilograms (i.e. Daltons - atomic mass units)
- The unit of charge (q_o) is $1.60217733 \times 10^{-19}$ coulombs (i.e. electrons - units of proton charge)
- The unit of energy ($E_o = m_o(\ell_o/t_o)^2$) is $1.6605402 \times 10^{-23}$ Joules (10 J mol^{-1})
- The unit of pressure ($\mathcal{P}_o = E_o\ell_o^{-3}$) is 1.6605402×10^7 Pascals (163.882576 atmospheres)
- Planck's constant ($\hbar$) which is $6.350780668 \times E_o t_o$.

In addition, the following conversion factors are used:

- The coulombic conversion factor (γ_o) is:

$$\gamma_o = \frac{1}{E_o}\left[\frac{q_o^2}{4\pi\epsilon_o\ell_o}\right] = 138935.4835 \quad ,$$

such that:

$$U_{MKS} = E_o\gamma_o U_{Internal} \quad ,$$

where U represents the configuration energy.

- The Boltzmann factor (k_B) is 0.831451115 E_oK^{-1}, such that:

$$T = E_{kin}/k_B$$

represents the conversion from kinetic energy (in internal units) to temperature (in Kelvin).

Note: In the DL_POLY_3 OUTPUT file, the print out of pressure is in units of kilo-atmospheres at all times. The unit of energy is either DL_POLY_3 units specified above, or in other units specified by the user at run time. The default is DL_POLY units.

1.3.9 Error Messages

All errors detected by DL_POLY_3 during run time initiate a call to the subroutine ERROR, which prints an error message in the standard output file and terminates the program. All terminations of the program are global (i.e. every node of the parallel computer will be informed of the termination condition and stop executing).

In addition to terminal error messages, DL_POLY_3 will sometimes print warning messages. These indicate that the code has detected something that is unusual or inconsistent. The detection is non-fatal, but the user should make sure that the warning does represent a harmless condition.

1.4 The DL_POLY_3 Directory Structure

The entire DL_POLY_3 package is stored in a Unix directory structure. The topmost directory is named *dl_poly_3.nn*, where *nn* is a generation number. Beneath this directory are several sub-directories named: *source*, *utility*, *data*, *bench*, *execute*, *build*, *public*, and *java*.

Briefly, the content of each sub-directory is as follows:

sub-directory	contents
source	primary subroutines for the DL_POLY_3 package
utility	subroutines, programs and example data for all utilities
data	example input and output files for DL_POLY_3
bench	large test cases suitable for benchmarking
execute	the DL_POLY_3 run-time directory
build	makefiles to assemble and compile DL_POLY_3 programs
public	directory of routines donated by DL_POLY_3 users
java	directory of Java and FORTRAN routines for the Java GUI.

A more detailed description of each sub-directory follows.

1.4.1 The *source* Sub-directory

In this sub-directory all the essential source code for DL_POLY_3 , excluding the utility software is stored. In keeping with the 'package' concept of DL_POLY_3 , it does not contain any complete programs; these are assembled at compile time using an appropriate makefile. The subroutines in this sub-directory are documented in Chapter 6.

1.4.2 The *utility* Sub-directory

This sub-directory stores all the utility subroutines, functions and programs in DL_POLY_3 , together with examples of data. The various routines in this sub-directory are documented in the DL_POLY_2 Reference Manual. Users who devise their own utilities are advised to store them in the *utility* sub-directory.

1.4.3 The *data* Sub-directory

This sub-directory contains examples of input and output files for testing the released version of DL_POLY_3 . The examples of input data are copied into the *execute* sub-directory when a program is being tested. The test cases are documented in Chapter 7.

1.4.4 The *bench* Sub-directory

This directory contains examples of input and output data for DL_POLY_3 that are suitable for benchmarking DL_POLY_3 on large scale computers. These are described in Chapter 7.

1.4.5 The *execute* Sub-directory

In the supplied version of DL_POLY_3 , this sub-directory contains only a few macros for copying and storing data from and to the *data* sub-directory and for submitting programs for execution (see Appendix B). However when a DL_POLY_3 program is assembled using its makefile, it will be placed in this sub-directory and will subsequently be executed from here. The output from the job will also appear here, so users will find it convenient to use this sub-directory if they wish to use DL_POLY_3 as intended. (The experienced user is not absolutely required to use DL_POLY_3 this way however.)

1.4.6 The *build* Sub-directory

This sub-directory contains the standard makefiles for the creation (i.e. compilation and linking) of the DL_POLY_3 simulation program. The makefiles supplied select the appropriate subroutines from the *source* sub-directory and deposit the executable program in the *execute* directory. The user is advised to copy the appropriate makefile into the *source* directory, in case any modifications are required. The copy in the *build* sub-directory will then serve as a backup.

1.4.7 The *public* Sub-directory

This sub-directory contains assorted routines donated by DL_POLY users. Potential users should note that these routines are **unsupported** and come **without any guarantee or liability what-**

soever. They should be regarded as potentially useful resources to be hacked into shape as needed by the user. This directory is available from the CCP5 Program Library by direct FTP(see below).

1.4.8 The *java* Sub-directory

The DL_POLY_3 Java Graphical User Interface (GUI) is based on the Java language developed by Sun. The Java source code for this GUI is to be found in this sub-directory, along with a few FORTRAN sub-sub-directories which contain some additional capabilities accessible from the GUI. These sources are complete and sufficient to create a working GUI, provided the user has installed the Java Development Kit, (1.3 or above) which is available free from Sun at

http://java.sun.com .

The GUI, once compiled, may be executed on any machine where Java is installed, though note the FORTRAN components will need to be recompiled if the machine is changed [17].

1.5 Obtaining the Source Code

To obtain a copy of DL_POLY_3 it is first necessary to obtain a licence from Daresbury Laboratory. A copy of the licence form may be obtained in two ways: either by selecting the licence button on the World Wide Web site:

http://www.cse.clrc.ac.uk/msi/software/DL_POLY

and downloading and printing the file, or by using FTP to copy the postscript file from the CCP5 Program Library at Daresbury Laboratory in the following manner:

1. move to the desired directory on YOUR machine,
2. type: **ftp ftp.dl.ac.uk**
3. enter userid: **anonymous**
4. enter passwd: (use your name and site)
5. change to the directory: **cd ccp5/DL_POLY/DOCUMENTS**
6. type: **binary**
7. type: **get LICENCE.ps.Z** or **get GROUP.LICENCE.ps.Z**
8. type: **quit**

The licence file will need to be uncompressed (using the unix *uncompress* command) before printing. Note that there are two versions of the licence available; one for single academic users (with perhaps one or two postgraduate students); and one for academic groups (with perhaps several research staff, including postdoctoral and permanent staff). Choose the one most suitable for you. The licences are *package licences* which allow you access to all the DL_POLY software.

Once you have obtained the licence form, you should sign it and return it to the following address:

```
Dr. W. Smith
DL_POLY Program Library
Computational Science and Engineering Department
CCLRC Daresbury Laboratory
Daresbury
Warrington WA4 4AD
UK
```

Please return the licence **by post please**. When the signed licence has been received the DL_POLY_3 source code will be transferred by ftp. We will need to contact you about this procedure, so **please supply your e-mail address**. Please note we cannot create accounts on any of our machines for this purpose.

The DL_POLY_3 User Manual is freely available via World Wide Web or ftp, in the same manner as the licence form. The much larger DL_POLY_2 Reference Manual is also available.

The *bench* and *public* subdirectories of DL_POLY_3 are not issued in the standard package, but can be downloaded directly by FTP from FTP site (in the *ccp5/DL_POLY/DL_POLY_3* directory) as described above.

Note: Daresbury Laboratory is the sole centre for the distribution of DL_POLY_3 and copies obtained from elsewhere will be regarded as illegal and will not be supported.

1.6 Other Information

The DL_POLY_3 web site:

http : //www.cse.clrc.ac.uk/msi/software/DL_POLY

provides additional information in the form of

1. Access to all documentation (including licences)
2. Frequently asked questions
3. Bug reports.

Daresbury Laboratory also maintains two DL_POLY associated electronic mailing lists:

1. *dl_poly_news* - to which all registered DL_POLY_3 users are automatically subscribed. It is via this list that error reports and announcements of new versions are made. If you are a DL_POLY_3 user, but not on this list you may request to be added. Contact w.smith@dl.ac.uk
2. *dl_poly_mail* - is a group list which is available to DL_POLY_3 users by request. Its purpose is to allow DL_POLY_3 users to broadcast information and queries to each other. To subscribe to this list send a mail message to majordomo@dl.ac.uk with the one-line message:

   subscribe dl_poly_mail

   Subsequent messages may be broadcast by e-mailing to: dl_poly_mail@dl.ac.uk. Note that this is a vetted list, so electronic spam is not possible.

Chapter 2

Force Fields

Scope of Chapter

This chapter describes the variety of interaction potentials available in DL_POLY_3 .

2.1 Introduction to DL_POLY_3 Force Field

The force field is the set of functions needed to define the interactions in a molecular system. These may have a wide variety of analytical forms, with some basis in chemical physics, which must be parameterised to give the correct energy and forces. A huge variety of forms is possible and for this reason the DL_POLY_3 force field is designed to be adaptable. While it is not supplied with its own force field parameters, many of the functions familiar to GROMOS [14], Dreiding [15] and AMBER [16] users have been coded in the package, as well as less familiar forms. In addition DL_POLY_3 retains the possibility of the user defining additional potentials.

In DL_POLY_3 the total configuration energy of a molecular system may be written as:

$$\begin{aligned} U(\underline{r}_1,\underline{r}_2,\ldots,\underline{r}_N) \;=\; & \sum_{i_{shel}=1}^{N_{shel}} U_{shel}(i_{shel},\underline{r}_{core},\underline{r}_{shell}) \\ & + \sum_{i_{teth}=1}^{N_{teth}} U_{teth}(i_{teth},\underline{r}_i(\sqcup=t),\underline{r}_i(\sqcup=0)) \\ & + \sum_{i_{bond}=1}^{N_{bond}} U_{bond}(i_{bond},\underline{r}_a,\underline{r}_b) \\ & + \sum_{i_{angl}=1}^{N_{angl}} U_{angl}(i_{angl},\underline{r}_a,\underline{r}_b,\underline{r}_c) \\ & + \sum_{i_{dihd}=1}^{N_{dihd}} U_{dihd}(i_{dihd},\underline{r}_a,\underline{r}_b,\underline{r}_c,\underline{r}_d) \\ & + \sum_{i_{inv}=1}^{N_{inv}} U_{inv}(i_{inv},\underline{r}_a,\underline{r}_b,\underline{r}_c,\underline{r}_d) \\ & + \sum_{i=1}^{N-1}\sum_{j>i}^{N} U_{2_body}^{(metal)}(i,j,|\underline{r}_i-\underline{r}_j|) \\ & + \sum_{i=1}^{N}\sum_{j\neq i}^{N}\sum_{k\neq j}^{N} U_{tersoff}(i,j,k,\underline{r}_i,\underline{r}_j,\underline{r}_k) \\ & + \sum_{i=1}^{N-2}\sum_{j>i}^{N-1}\sum_{k>j}^{N} U_{3_body}(i,j,k,\underline{r}_i,\underline{r}_j,\underline{r}_k) \\ & + \sum_{i=1}^{N-3}\sum_{j>i}^{N-2}\sum_{k>j}^{N-1}\sum_{n>k}^{N} U_{4_body}(i,j,k,n,\underline{r}_i,\underline{r}_j,\underline{r}_k,\underline{r}_n) \\ & + \sum_{i=1}^{N} U_{extn}(i,\underline{r}_i,\underline{v}_i) \quad , \end{aligned} \tag{2.1}$$

where U_{shel}, U_{teth}, U_{bond}, U_{angl}, U_{dihd}, U_{inv}, U_{pair}, $U_{tersoff}$, U_{3_body} and U_{4_body} are empirical interaction functions representing ion core-shell polarisation, tethered particles, chemical bonds, valence angles, dihedral (and improper dihedral angles), inversion angles, two-body, Tersoff, three-body and four-body forces respectively. The first six are regarded by DL_POLY_3 as *intra*-molecular interactions and the next four as *inter*-molecular interactions. The final term U_{extn} represents an *external field* potential. The position vectors $\underline{r}_a, \underline{r}_b, \underline{r}_c$ and $\underline{r}_d$ refer to the positions of the atoms specifically involved in a given interaction. (Almost universally, it is the *differences* in position

that determine the interaction.) The numbers N_{shel}, N_{teth}, N_{bond}, N_{angl}, N_{dihd} and N_{inv} refer to the total numbers of these respective interactions present in the simulated system, and the indices i_{shel}, i_{teth}, i_{bond}, i_{angl}, i_{dihd} and i_{inv} uniquely specify an individual interaction of each type. It is important to note that there is no global specification of the intramolecular interactions in DL_POLY_3 - all core-shell units, tethered particles, chemical bonds, valence angles, dihedral angles and inversion angles must be individually cited. The same applies for bond constraints and pmf constraints.

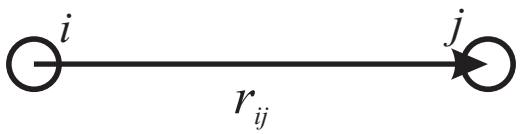
The indices i, j (and k, n) appearing in the intermolecular interactions' (non-bonded) terms indicate the atoms involved in the interaction. There is normally a very large number of these and they are therefore specified globally according to the atom *types* involved rather than indices. In DL_POLY_3 it is assumed that the "pure" two-body terms arise from van der Waals interactions (regarded as short-ranged) and electrostatic interactions (coulombic, also regarded as long-ranged). Long-ranged forces require special techniques to evaluate accurately (see Section 2.4). The metal forces are many-body forces which are functionally presented in an expansion of many two-body contributions (and so evaluated in the two_body routines). In DL_POLY_3 the three-body terms are restricted to valence angle and H-bond forms.

Throughout this chapter the description of the force field assumes the simulated system is described as an assembly of atoms. This is for convenience only, and readers should understand that DL_POLY_3 does recognize molecular entities, defined through constraint bonds. These matters are discussed in greater detail in Section 3.2.

2.2 The Intramolecular Potential Functions

In this section we catalogue and describe the forms of potential function available in DL_POLY_3 . The **keywords** required to select potential forms are given in brackets () before each definition. The derivations of the atomic forces, virial and stress tensor are also outlined.

2.2.1 Bond Potentials



The interatomic bond vector.

The bond potentials describe *explicit* chemical bonds between specified atoms. They are all functions of the interatomic distance. Only the coulomb potential makes an exception as it depends on the charges of the specified atoms. The potential functions available are as follows:

1. Harmonic bond: (**harm**)
$$U(r_{ij}) = \frac{1}{2}k(r_{ij} - r_o)^2 \tag{2.2}$$

2. Morse potential: (**mors**)
$$U(r_{ij}) = E_o[\{1 - \exp(-k(r_{ij} - r_o))\}^2 - 1] \tag{2.3}$$

3. 12-6 potential bond: (**12-6**)

$$U(r_{ij}) = \left(\frac{A}{r_{ij}^{12}}\right) - \left(\frac{B}{r_{ij}^{6}}\right) \tag{2.4}$$

4. Restrained harmonic: (**rhrm**)

$$U(r_{ij}) = \begin{cases} \frac{1}{2}k(r_{ij}-r_o)^2 & : \quad |r_{ij}-r_o| \le r_c \\ \frac{1}{2}kr_c^2 + kr_c(|r_{ij}-r_o|-r_c) & : \quad |r_{ij}-r_o| > r_c \end{cases} \tag{2.5}$$

5. Quartic potential: (**quar**)

$$U(r_{ij}) = \frac{k}{2}(r_{ij}-r_o)^2 + \frac{k'}{3}(r_{ij}-r_o)^3 + \frac{k''}{4}(r_{ij}-r_o)^4 \tag{2.6}$$

6. Buckingham potential: (**buck**)

$$U(r_{ij}) = A\ \exp\left(-\frac{r_{ij}}{\rho}\right) - \frac{C}{r_{ij}^6} \tag{2.7}$$

7. Coulomb potential: (**coul**)

$$U(r_{ij}) = \frac{k}{4\pi\epsilon_0\epsilon}\frac{q_i q_j}{r_{ij}} \quad , \tag{2.8}$$

   where q_ℓ is the charge on an atom labelled ℓ.

In these formulae r_{ij} is the distance between atoms labelled i and j:

$$r_{ij} = |\underline{r}_j - \underline{r}_i| \quad , \tag{2.9}$$

where $\underline{r}_\ell$ is the position vector of an atom labelled ℓ. [1]

The force on the atom j arising from a bond potential is obtained using the general formula:

$$\underline{f}_j = -\frac{1}{r_{ij}}\left[\frac{\partial}{\partial r_{ij}}U(r_{ij})\right]\underline{r}_{ij} \quad . \tag{2.10}$$

The force $\underline{f}_i$ acting on atom i is the negative of this.

The contribution to be added to the atomic virial is given by

$$\mathcal{W} = -\underline{r}_{ij}\cdot\underline{f}_j \quad , \tag{2.11}$$

with only *one* such contribution from each bond.

The contribution to be added to the atomic stress tensor is given by

$$\sigma^{\alpha\beta} = r_{ij}^{\alpha} f_j^{\beta} \quad , \tag{2.12}$$

where α and β indicate the x, y, z components. The atomic stress tensor derived in this way is symmetric.

In DL_POLY_3 bond forces are handled by the routine BONDS_FORCES.

[1] Note: some DL_POLY_3 routines may use the convention that $\underline{r_{ij}} = \underline{r}_i - \underline{r}_j$.

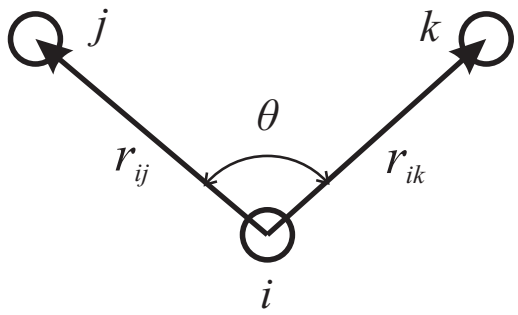
2.2.2 Distance Restraints

In DL_POLY_3 distance restraints, in which the separation between two atoms, is maintained around some preset value r_0 is handled as a special case of bond potentials. As a consequence, distance restraints may be applied only between atoms in the same molecule. Unlike with application of the "pure" bond potentials, the electrostatic and van der Waals interactions between the pair of atoms are still evaluated when distance restraints are applied. All the potential forms of the previous section are avaliable as distance restraints, although they have different key words:

1. Harmonic potential: (**-hrm**)
2. Morse potential: (**-mrs**)
3. 12-6 potential bond: (**-126**)
4. Restrained harmonic: (**-rhm**)
5. Quartic potential: (**-qur**)
6. Buckingham potential: (**-bck**)
7. Coulomb potential: (**-cul**)

In DL_POLY_3 distance restraints are handled by the routine BONDS_FORCES.

2.2.3 Valence Angle Potentials



The valence angle and associated vectors

The valence angle potentials describe the bond bending terms between the specified atoms. They should not be confused with the three-body potentials described later, which are defined by atom types rather than indices.

1. Harmonic: (**harm**)
$$U(\theta_{jik}) = \frac{k}{2}(\theta_{jik} - \theta_0)^2 \tag{2.13}$$
2. Quartic: (**quar**)
$$U(\theta_{jik}) = \frac{k}{2}(\theta_{jik} - \theta_0)^2 + \frac{k'}{3}(\theta_{jik} - \theta_0)^3 + \frac{k''}{4}(\theta_{jik} - \theta_0)^4 \tag{2.14}$$

3. Truncated harmonic: (**thrm**)

$$U(\theta_{jik}) = \frac{k}{2}(\theta_{jik} - \theta_0)^2 \exp[-(r_{ij}^8 + r_{ik}^8)/\rho^8] \tag{2.15}$$

4. Screened harmonic: (**shrm**)

$$U(\theta_{jik}) = \frac{k}{2}(\theta_{jik} - \theta_0)^2 \exp[-(r_{ij}/\rho_1 + r_{ik}/\rho_2)] \tag{2.16}$$

5. Screened Vessal [25]: (**bvs1**)

$$\begin{aligned} U(\theta_{jik}) &= \frac{k}{8(\theta_{jik} - \pi)^2}\left[(\theta_0 - \pi)^2 - (\theta_{jik} - \pi)^2\right]^2 \times \\ &\quad \exp[-(r_{ij}/\rho_1 + r_{ik}/\rho_2)] \end{aligned} \tag{2.17}$$

6. Truncated Vessal [26]: (**bvs2**)

$$\begin{aligned} U(\theta_{jik}) &= k\ [\ \theta_{jik}^a(\theta_{jik} - \theta_0)^2(\theta_{jik} + \theta_0 - 2\pi)^2 - \\ &\quad \frac{a}{2}\pi^{a-1}(\theta_{jik} - \theta_0)^2(\pi - \theta_0)^3\ ]\ \ \exp[-(r_{ij}^8 + r_{ik}^8)/\rho^8] \end{aligned} \tag{2.18}$$

7. Harmonic cosine: (**hcos**)

$$U(\theta_{jik}) = \frac{k}{2}(cos(\theta_{jik}) - cos(\theta_0))^2 \tag{2.19}$$

8. Cosine: (**cos**)

$$U(\theta_{jik}) = A\ [1 + cos(m\theta_{jik} - \delta)] \tag{2.20}$$

9. MM3 stretch-bend [27]: (**mmsb**)

$$U(\theta_{jik}) = A\ (\theta_{jik} - \theta_0)\ (r_{ij} - r_{ij}^o)\ (r_{ik} - r_{ik}^o) \tag{2.21}$$

In these formulae θ_{jik} is the angle between bond vectors $\underline{r}_{ij}$ and $\underline{r}_{ik}$:

$$\theta_{jik} = cos^{-1}\left\{\frac{\underline{r}_{ij} \cdot \underline{r}_{ik}}{r_{ij} r_{ik}}\right\} \quad . \tag{2.22}$$

In DL_POLY_3 the most general form for the valence angle potentials can be written as:

$$U(\theta_{jik}, r_{ij}, r_{ik}) = A(\theta_{jik})S(r_{ij})S(r_{ik}) \quad , \tag{2.23}$$

where $A(\theta)$ is a purely angular function and $S(r)$ is a screening or truncation function. All the function arguments are scalars. With this reduction the force on an atom derived from the valence angle potential is given by:

$$f_\ell^\alpha = -\frac{\partial}{\partial r_\ell^\alpha} U(\theta_{jik}, r_{ij}, r_{ik}) \quad , \tag{2.24}$$

with atomic label ℓ being one of i, j, k and α indicating the x, y, z component. The derivative is

$$\begin{aligned} -\frac{\partial}{\partial r_\ell^\alpha} U(\theta_{jik}, r_{ij}, r_{ik}) &= -S(r_{ij})S(r_{ik})\frac{\partial}{\partial r_\ell^\alpha} A(\theta_{jik}) \\ &\quad -A(\theta_{jik})S(r_{ik})(\delta_{\ell j} - \delta_{\ell i})\frac{r_{ij}^\alpha}{r_{ij}}\frac{\partial}{\partial r_{ij}} S(r_{ij}) \\ &\quad -A(\theta_{jik})S(r_{ij})(\delta_{\ell k} - \delta_{\ell i})\frac{r_{ik}^\alpha}{r_{ik}}\frac{\partial}{\partial r_{ik}} S(r_{ik}) \quad , \end{aligned} \tag{2.25}$$

with $\delta_{ab} = 1$ if $a = b$ and $\delta_{ab} = 0$ if $a \neq b$. In the absence of screening terms $S(r)$, this formula reduces to:

$$-\frac{\partial}{\partial r_\ell^\alpha} U(\theta_{jik}, r_{ij}, r_{ik}) = -\frac{\partial}{\partial r_\ell^\alpha} A(\theta_{jik}) \quad . \tag{2.26}$$

The derivative of the angular function is

$$-\frac{\partial}{\partial r_\ell^\alpha} A(\theta_{jik}) = \left\{ \frac{1}{\sin(\theta_{jik})} \right\} \frac{\partial}{\partial \theta_{jik}} A(\theta_{jik}) \frac{\partial}{\partial r_\ell^\alpha} \left\{ \frac{\underline{r}_{ij} \cdot \underline{r}_{ik}}{r_{ij} r_{ik}} \right\} \quad , \tag{2.27}$$

with

$$\begin{aligned} \frac{\partial}{\partial r_\ell^\alpha} \left\{ \frac{\underline{r}_{ij} \cdot \underline{r}_{ik}}{r_{ij} r_{ik}} \right\} \quad = \quad & (\delta_{\ell j} - \delta_{\ell i}) \frac{r_{ik}^\alpha}{r_{ij} r_{ik}} + (\delta_{\ell k} - \delta_{\ell i}) \frac{r_{ij}^\alpha}{r_{ij} r_{ik}} - \\ & \cos(\theta_{jik}) \left\{ (\delta_{\ell j} - \delta_{\ell i}) \frac{r_{ij}^\alpha}{r_{ij}^2} + (\delta_{\ell k} - \delta_{\ell i}) \frac{r_{ik}^\alpha}{r_{ik}^2} \right\} \quad . \end{aligned} \tag{2.28}$$

The atomic forces are then completely specified by the derivatives of the particular functions $A(\theta)$ and $S(r)$.

The contribution to be added to the atomic virial is given by

$$\mathcal{W} = -(\underline{r}_{ij} \cdot \underline{f}_j + \underline{r}_{ik} \cdot \underline{f}_k) \quad . \tag{2.29}$$

It is worth noting that in the absence of screening terms S(r), the virial is zero [28].

The contribution to be added to the atomic stress tensor is given by

$$\sigma^{\alpha\beta} = r_{ij}^\alpha f_j^\beta + r_{ik}^\alpha f_k^\beta \tag{2.30}$$

and the stress tensor is symmetric.

In DL_POLY_3 valence forces are handled by the routine ANGLES_FORCES.

2.2.4 Angular Restraints

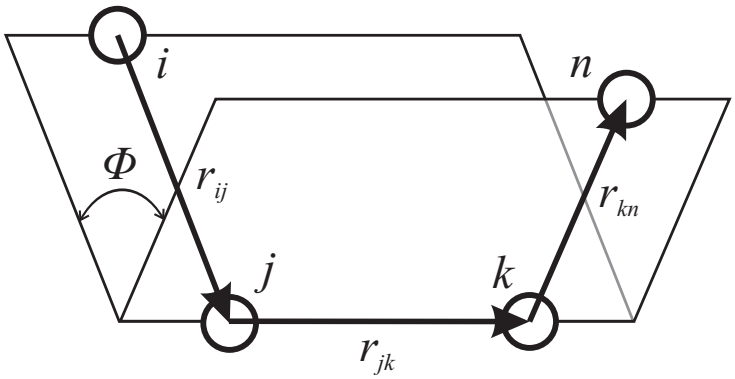
In DL_POLY_3 angle restraints, in which the angle subtended by a triplet of atoms, is maintained around some preset value θ_0 is handled as a special case of angle potentials. As a consequence angle restraints may be applied only between atoms in the same molecule. Unlike with application of the "pure" angle potentials, the electrostatic and van der Waals interactions between the pair of atoms are still evaluated when distance restraints are applied. All the potential forms of the previous section are avaliable as angular restraints, although they have different key words:

1. Harmonic: (**-hrm**)
2. Quartic: (**-qur**)
3. Truncated harmonic: (**-thm**)
4. Screened harmonic: (**-shm**)
5. Screened Vessal [25]: (**-bv1**)
6. Truncated Vessal [26]: (**-bv2**)

7. Harmonic cosine: (**-hcs**)
8. Cosine: (**-cos**)
9. MM3 stretch-bend [27]: (**-msb**)

In DL_POLY_3 angular restraints are handled by the routine ANGLES_FORCES.

2.2.5 Dihedral Angle Potentials



The dihedral angle and associated vectors

The dihedral angle potentials describe the interaction arising from torsional forces in molecules. (They are sometimes referred to as torsion potentials.) They require the specification of four atomic positions. The potential functions available in DL_POLY_3 are as follows:

1. Cosine potential: (**cos**)
$$U(\phi_{ijkn}) = A\ [1 + \cos(m\phi_{ijkn} - \delta)] \tag{2.31}$$
2. Harmonic: (**harm**)
$$U(\phi_{ijkn}) = \frac{k}{2}\ (\phi_{ijkn} - \phi_0)^2 \tag{2.32}$$
3. Harmonic cosine: (**hcos**)
$$U(\phi_{ijkn}) = \frac{k}{2}\ (\cos(\phi_{ijkn}) - \cos(\phi_0))^2 \tag{2.33}$$
4. Triple cosine: (**cos3**)
$$U(\phi) = \frac{1}{2}\ \{A_1\ (1 + \cos(\phi)) + A_2\ (1 - \cos(2\phi)) + A_3\ (1 + \cos(3\phi))\} \tag{2.34}$$
5. Ryckaert-Bellemans [29] with fixed constants a-f: (**ryck**)
$$U(\phi) = A\ \{\ a + b\ \cos(\phi) + c\ \cos^2(\phi) + d\ \cos^3(\phi) + e\ \cos^4(\phi) + f\ \cos^5(\phi)\ \} \tag{2.35}$$
6. Fluorinated Ryckaert-Bellemans [30] with fixed constants a-h: (**rbf**)
$$U(\phi) = A\ \{\ a + b\ \cos(\phi) + c\ \cos^2(\phi) + d\ \cos^3(\phi) + e\ \cos^4(\phi) + f\ \cos^5(\phi) + g\ \exp(-h(\phi - \pi)^2))\ \} \tag{2.36}$$

7. OPLS torsion potential: (**opls**)

$$U(\phi) = A_0 + \frac{1}{2}\left\{A_1\ (1+\cos(\phi)) + A_2\ (1-\cos(2\phi)) + A_3\ (1+\cos(4\phi))\right\} \tag{2.37}$$

In these formulae ϕ_{ijkn} is the dihedral angle defined by

$$\phi_{ijkn} = \cos^{-1}\{B(\underline{r}_{ij}, \underline{r}_{jk}, \underline{r}_{kn})\} \quad , \tag{2.38}$$

with

$$B(\underline{r}_{ij}, \underline{r}_{jk}, \underline{r}_{kn}) = \left\{ \frac{(\underline{r}_{ij} \times \underline{r}_{jk}) \cdot (\underline{r}_{jk} \times \underline{r}_{kn})}{|\underline{r}_{ij} \times \underline{r}_{jk}||\underline{r}_{jk} \times \underline{r}_{kn}|} \right\} \quad . \tag{2.39}$$

With this definition, the sign of the dihedral angle is positive if the vector product $(\underline{r}_{ij} \times \underline{r}_{jk}) \times (\underline{r}_{jk} \times \underline{r}_{kn})$ is in the same direction as the bond vector $\underline{r}_{jk}$ and negative if in the opposite direction.

The force on an atom arising from the dihedral potential is given by

$$f_\ell^\alpha = -\frac{\partial}{\partial r_\ell^\alpha} U(\phi_{ijkn}) \quad , \tag{2.40}$$

with ℓ being one of i, j, k, n and α one of x, y, z. This may be expanded into

$$-\frac{\partial}{\partial r_\ell^\alpha} U(\phi_{ijkn}) = \left\{ \frac{1}{\sin(\phi_{ijkn})} \right\} \frac{\partial}{\partial \phi_{ijkn}} U(\phi_{ijkn}) \frac{\partial}{\partial r_\ell^\alpha} B(\underline{r}_{ij}, \underline{r}_{jk}, \underline{r}_{kn}) \quad . \tag{2.41}$$

The derivative of the function $B(\underline{r}_{ij}, \underline{r}_{jk}, \underline{r}_{kn})$ is

$$\begin{aligned} \frac{\partial}{\partial r_\ell^\alpha} B(\underline{r}_{ij}, \underline{r}_{jk}, \underline{r}_{kn}) \ &= \ \frac{1}{|\underline{r}_{ij} \times \underline{r}_{jk}||\underline{r}_{jk} \times \underline{r}_{kn}|} \frac{\partial}{\partial r_\ell^\alpha} \{(\underline{r}_{ij} \times \underline{r}_{jk}) \cdot (\underline{r}_{jk} \times \underline{r}_{kn})\} \ - \\ & \frac{\cos(\phi_{ijkn})}{2} \left\{ \frac{1}{|\underline{r}_{ij} \times \underline{r}_{jk}|^2} \frac{\partial}{\partial r_\ell^\alpha} |\underline{r}_{ij} \times \underline{r}_{jk}|^2 + \frac{1}{|\underline{r}_{jk} \times \underline{r}_{kn}|^2} \frac{\partial}{\partial r_\ell^\alpha} |\underline{r}_{jk} \times \underline{r}_{kn}|^2 \right\} \quad , \end{aligned} \tag{2.42}$$

with

$$\begin{aligned} \frac{\partial}{\partial r_\ell^\alpha} \{(\underline{r}_{ij} \times \underline{r}_{jk}) \cdot (\underline{r}_{jk} \times \underline{r}_{kn})\} \ &= \ r_{ij}^\alpha ([\underline{r}_{jk}\underline{r}_{jk}]_\alpha (\delta_{\ell k} - \delta_{\ell n}) + [\underline{r}_{jk}\underline{r}_{kn}]_\alpha (\delta_{\ell k} - \delta_{\ell j})) + \\ & \quad r_{jk}^\alpha ([\underline{r}_{ij}\underline{r}_{jk}]_\alpha (\delta_{\ell n} - \delta_{\ell k}) + [\underline{r}_{jk}\underline{r}_{kn}]_\alpha (\delta_{\ell j} - \delta_{\ell i})) + \\ & \quad r_{kn}^\alpha ([\underline{r}_{ij}\underline{r}_{jk}]_\alpha (\delta_{\ell k} - \delta_{\ell j}) + [\underline{r}_{jk}\underline{r}_{jk}]_\alpha (\delta_{\ell i} - \delta_{\ell j})) + \\ & \quad 2 r_{jk}^\alpha [\underline{r}_{ij}\underline{r}_{kn}]_\alpha (\delta_{\ell j} - \delta_{\ell k}) \quad , \end{aligned} \tag{2.43}$$

$$\begin{aligned} \frac{\partial}{\partial r_\ell^\alpha} |\underline{r}_{ij} \times \underline{r}_{jk}|^2 \ &= \ 2 r_{ij}^\alpha ([\underline{r}_{jk}\underline{r}_{jk}]_\alpha (\delta_{\ell j} - \delta_{\ell i}) + [\underline{r}_{ij}\underline{r}_{jk}]_\alpha (\delta_{\ell j} - \delta_{\ell k})) + \\ & \quad 2 r_{jk}^\alpha ([\underline{r}_{ij}\underline{r}_{ij}]_\alpha (\delta_{\ell k} - \delta_{\ell j}) + [\underline{r}_{ij}\underline{r}_{jk}]_\alpha (\delta_{\ell i} - \delta_{\ell j})) \quad , \end{aligned} \tag{2.44}$$

$$\begin{aligned} \frac{\partial}{\partial r_\ell^\alpha} |\underline{r}_{jk} \times \underline{r}_{kn}|^2 \ &= \ 2 r_{kn}^\alpha ([\underline{r}_{jk}\underline{r}_{jk}]_\alpha (\delta_{\ell n} - \delta_{\ell k}) + [\underline{r}_{jk}\underline{r}_{kn}]_\alpha (\delta_{\ell j} - \delta_{\ell k})) + \\ & \quad 2 r_{jk}^\alpha ([\underline{r}_{kn}\underline{r}_{kn}]_\alpha (\delta_{\ell k} - \delta_{\ell j}) + [\underline{r}_{jk}\underline{r}_{kn}]_\alpha (\delta_{\ell k} - \delta_{\ell n})) \quad . \end{aligned} \tag{2.45}$$

Where we have used the the following definition:

$$[\underline{a}\ \underline{b}]_\alpha = \sum_\beta (1-\delta_{\alpha\beta}) a^\beta b^\beta \quad . \tag{2.46}$$

Formally, the contribution to be added to the atomic virial is given by

$$\mathcal{W} = -\sum_{i=1}^{4} \underline{r}_i \cdot \underline{f}_i \quad . \tag{2.47}$$

However, it is possible to show (by tedious algebra using the above formulae, or more elegantly by thermodynamic arguments [28],) that the dihedral makes *no* contribution to the atomic virial.

The contribution to be added to the atomic stress tensor is given by

$$\begin{aligned} \sigma^{\alpha\beta} &= r^\alpha_{ij} p^\beta_i + r^\alpha_{jk} p^\beta_{jk} + r^\alpha_{kn} p^\beta_n \\ &\quad -\frac{\cos(\phi_{ijkn})}{2}\left\{ r^\alpha_{ij} g^\beta_i + r^\alpha_{jk} g^\beta_k + r^\alpha_{jk} h^\beta_j + r^\alpha_{kn} h^\beta_n \right\} \quad , \end{aligned} \tag{2.48}$$

with

$$\begin{aligned} p^\alpha_i &= (r^\alpha_{jk}[\underline{r}_{jk}\underline{r}_{kn}]_\alpha - r^\alpha_{kn}[\underline{r}_{jk}\underline{r}_{jk}]_\alpha)/(|\underline{r}_{ij}\times\underline{r}_{jk}||\underline{r}_{jk}\times\underline{r}_{kn}|) \\ p^\alpha_n &= (r^\alpha_{jk}[\underline{r}_{ij}\underline{r}_{jk}]_\alpha - r^\alpha_{ij}[\underline{r}_{jk}\underline{r}_{jk}]_\alpha)/(|\underline{r}_{ij}\times\underline{r}_{jk}||\underline{r}_{jk}\times\underline{r}_{kn}|) \\ p^\alpha_{jk} &= (r^\alpha_{ij}[\underline{r}_{jk}\underline{r}_{kn}]_\alpha + r^\alpha_{kn}[\underline{r}_{ij}\underline{r}_{jk}]_\alpha - 2r^\alpha_{jk}[\underline{r}_{ij}\underline{r}_{kn}]_\alpha)/(|\underline{r}_{ij}\times\underline{r}_{jk}||\underline{r}_{jk}\times\underline{r}_{kn}|) \\ g^\alpha_i &= 2(r^\alpha_{ij}[\underline{r}_{jk}\underline{r}_{jk}]_\alpha - r^\alpha_{jk}[\underline{r}_{ij}\underline{r}_{jk}]_\alpha)/|\underline{r}_{ij}\times\underline{r}_{jk}|^2 \\ g^\alpha_k &= 2(r^\alpha_{jk}[\underline{r}_{ij}\underline{r}_{ij}]_\alpha - r^\alpha_{ij}[\underline{r}_{ij}\underline{r}_{jk}]_\alpha)/|\underline{r}_{ij}\times\underline{r}_{jk}|^2 \\ h^\alpha_j &= 2(r^\alpha_{jk}[\underline{r}_{kn}\underline{r}_{kn}]_\alpha - r^\alpha_{kn}[\underline{r}_{jk}\underline{r}_{kn}]_\alpha)/|\underline{r}_{jk}\times\underline{r}_{kn}|^2 \\ h^\alpha_n &= 2(r^\alpha_{kn}[\underline{r}_{kn}\underline{r}_{kn}]_\alpha - r^\alpha_{jk}[\underline{r}_{jk}\underline{r}_{kn}]_\alpha)/|\underline{r}_{jk}\times\underline{r}_{kn}|^2 \quad . \end{aligned} \tag{2.49}$$

The sum of the diagonal elements of the stress tensor is zero (since the virial is zero) and the matrix is symmetric.

Lastly, it should be noted that the above description does not take into account the possible inclusion of distance-dependent 1-4 interactions, as permitted by some force fields. Such interactions are permissible in DL_POLY_3 and are described in the section on pair potentials below. DL_POLY_3 also permits scaling of the 1-4 interactions by a numerical factor. 1-4 interactions do, of course, contribute to the atomic virial.

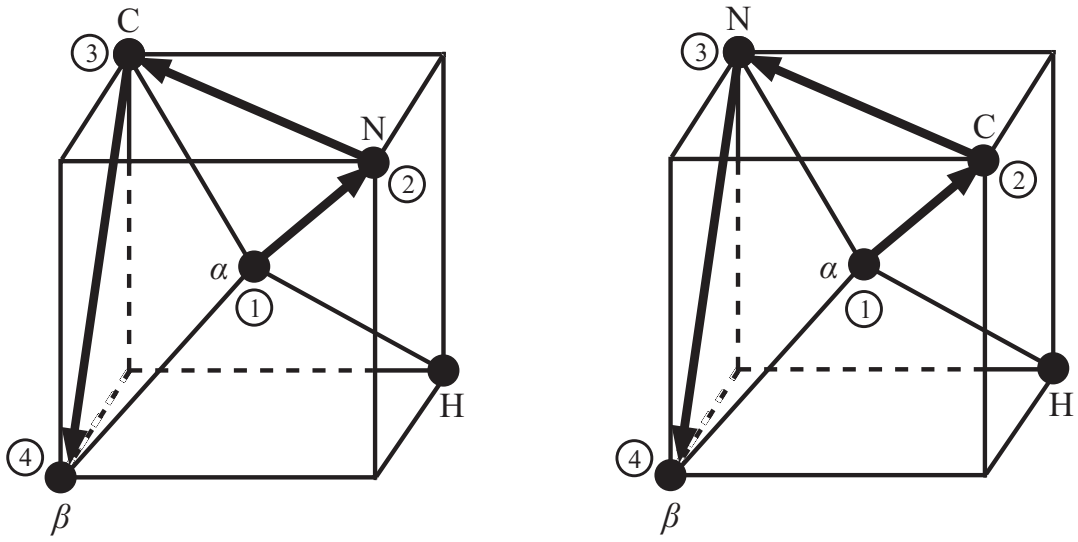
In DL_POLY_3 dihedral forces are handled by the routine DIHEDRALS_FORCES.

2.2.6 Improper Dihedral Angle Potentials

Improper dihedrals are used to restrict the geometry of molecules and as such need not have a simple relation to conventional chemical bonding. DL_POLY_3 makes no distinction between dihedral and improper dihedral angle functions (both are calculated by the same subroutines) and all the comments made in the preceding section apply.

An important example of the use of the improper dihedral is to conserve the structure of chiral centres in molecules modelled by united-atom centres. For example α-amino acids such as alanine ($CH_3CH(NH_2)COOH$), in which it is common to represent the CH_3 and CH groups as single centres. Conservation of the chirality of the α carbon is achieved by defining a harmonic improper dihedral

angle potential with an equilibrium angle of 35.264^o. The angle is defined by vectors $\underline{r}_{12}$, $\underline{r}_{23}$ and $\underline{r}_{34}$, where the atoms 1,2,3 and 4 are shown in the following figure. The figure defines the D and L enantiomers consistent with the international (IUPAC) convention. When defining the dihedral, the atom indices are entered in DL_POLY_3 in the order 1-2-3-4.



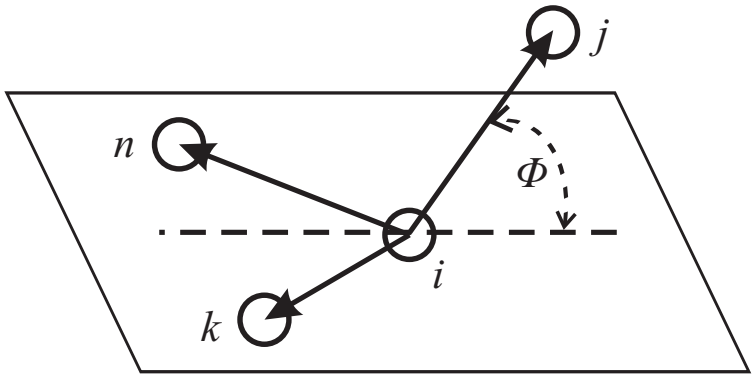
L = α - N - C - β
1 2 3 4

D = α - C - N - β
1 2 3 4

The L and D enantiomers and defining vectors

In DL_POLY_3 improper dihedral forces are handled by the routine DIHEDRALS_FORCES.

2.2.7 Inversion Angle Potentials



The inversion angle and associated vectors

The inversion angle potentials describe the interaction arising from a particular geometry of three atoms around a central atom. The best known example of this is the arrangement of hydrogen atoms around nitrogen in ammonia to form a trigonal pyramid. The hydrogens can 'flip' like an inverting umbrella to an alternative structure, which in this case is identical, but in principle causes

a change in chirality. The force restraining the ammonia to one structure can be described as an inversion potential (though it is usually augmented by valence angle potentials also). The inversion angle is defined in the figure above - **note that the inversion angle potential is a sum of the three possible inversion angle terms**. It resembles a dihedral potential in that it requires the specification of four atomic positions.

The potential functions available in DL_POLY_3 are as follows:

1. Harmonic: (**harm**)
$$U(\phi_{ijkn}) = \frac{k}{2}\ (\phi_{ijkn} - \phi_0)^2 \tag{2.50}$$

2. Harmonic cosine: (**hcos**)
$$U(\phi_{ijkn}) = \frac{k}{2}\ (\cos(\phi_{ijkn}) - \cos(\phi_0))^2 \tag{2.51}$$

3. Planar potential: (**plan**)
$$U(\phi_{ijkn}) = A\ \ [1 - \cos(\phi_{ijkn})] \tag{2.52}$$

In these formulae ϕ_{ijkn} is the inversion angle defined by

$$\phi_{ijkn} = \cos^{-1}\left\{\frac{\underline{r}_{ij}\cdot\underline{w}_{kn}}{r_{ij}w_{kn}}\right\}\quad , \tag{2.53}$$

with

$$\underline{w}_{kn} = (\underline{r}_{ij}\cdot\hat{\underline{u}}_{kn})\hat{\underline{u}}_{kn} + (\underline{r}_{ij}\cdot\hat{\underline{v}}_{kn})\hat{\underline{v}}_{kn} \tag{2.54}$$

and the unit vectors

$$\begin{aligned}\hat{\underline{u}}_{kn} &= (\hat{\underline{r}}_{ik} + \hat{\underline{r}}_{in})/|\hat{\underline{r}}_{ik} + \hat{\underline{r}}_{in}| \\ \hat{\underline{v}}_{kn} &= (\hat{\underline{r}}_{ik} - \hat{\underline{r}}_{in})/|\hat{\underline{r}}_{ik} - \hat{\underline{r}}_{in}|\quad .\end{aligned} \tag{2.55}$$

As usual, $\underline{r}_{ij} = \underline{r}_j - \underline{r}_i$ *etc.* and the hat $\hat{\underline{r}}$ indicates a *unit* vector in the direction of $\underline{r}$. The total inversion potential requires the calculation of three such angles, the formula being derived from the above using the cyclic permutation of the indices $j \to k \to n \to j$ *etc.*

Equivalently, the angle ϕ_{ijkn} may be written as

$$\phi_{ijkn} = \cos^{-1}\left\{\frac{[(\underline{r}_{ij}\cdot\hat{\underline{u}}_{kn})^2 + (\underline{r}_{ij}\cdot\hat{\underline{v}}_{kn})^2]^{1/2}}{r_{ij}}\right\}\quad . \tag{2.56}$$

Formally, the force on an atom arising from the inversion potential is given by

$$f_\ell^\alpha = -\frac{\partial}{\partial r_\ell^\alpha}U(\phi_{ijkn})\quad , \tag{2.57}$$

with ℓ being one of i, j, k, n and α one of x, y, z. This may be expanded into

$$\begin{aligned}-\frac{\partial}{\partial r_\ell^\alpha}U(\phi_{ijkn}) &= \left\{\frac{1}{\sin(\phi_{ijkn})}\right\}\frac{\partial}{\partial\phi_{ijkn}}U(\phi_{ijkn})\times \\ &\quad \frac{\partial}{\partial r_\ell^\alpha}\left\{\frac{[(\underline{r}_{ij}\cdot\hat{\underline{u}}_{kn})^2 + (\underline{r}_{ij}\cdot\hat{\underline{v}}_{kn})^2]^{1/2}}{r_{ij}}\right\}\quad .\end{aligned} \tag{2.58}$$

Following through, the (extremely tedious!) differentiation gives the result:

$$
\begin{aligned}
f_\ell^\alpha \;=\; & \left\{\frac{1}{\sin(\phi_{ijkn})}\right\}\frac{\partial}{\partial\phi_{ijkn}}U(\phi_{ijkn})\times \\
& \left\{-(\delta_{\ell j}-\delta_{\ell i})\frac{cos(\phi_{ijkn})}{r_{ij}^2}r_{ij}^\alpha+\frac{1}{r_{ij}w_{kn}}\left[(\delta_{\ell j}-\delta_{\ell i})\{(\underline{r}_{ij}\cdot\hat{\underline{u}}_{kn})\hat{u}_{kn}^\alpha+(\underline{r}_{ij}\cdot\hat{\underline{v}}_{kn})\hat{v}_{kn}^\alpha\}\right.\right. \\
& +(\delta_{\ell k}-\delta_{\ell i})\frac{\underline{r}_{ij}\cdot\hat{\underline{u}}_{kn}}{u_{kn}r_{ik}}\left\{r_{ij}^\alpha-(\underline{r}_{ij}\cdot\hat{\underline{u}}_{kn})\hat{u}_{kn}^\alpha-(\underline{r}_{ij}\cdot\underline{r}_{ik}-(\underline{r}_{ij}\cdot\hat{\underline{u}}_{kn})(\underline{r}_{ik}\cdot\hat{\underline{u}}_{kn}))\frac{r_{ik}^\alpha}{r_{ik}^2}\right\} \\
& +(\delta_{\ell k}-\delta_{\ell i})\frac{\underline{r}_{ij}\cdot\hat{\underline{v}}_{kn}}{v_{kn}r_{ik}}\left\{r_{ij}^\alpha-(\underline{r}_{ij}\cdot\hat{\underline{v}}_{kn})\hat{v}_{kn}^\alpha-(\underline{r}_{ij}\cdot\underline{r}_{ik}-(\underline{r}_{ij}\cdot\hat{\underline{v}}_{kn})(\underline{r}_{ik}\cdot\hat{\underline{v}}_{kn}))\frac{r_{ik}^\alpha}{r_{ik}^2}\right\} \\
& +(\delta_{\ell n}-\delta_{\ell i})\frac{\underline{r}_{ij}\cdot\hat{\underline{u}}_{kn}}{u_{kn}r_{in}}\left\{r_{ij}^\alpha-(\underline{r}_{ij}\cdot\hat{\underline{u}}_{kn})\hat{u}_{kn}^\alpha-(\underline{r}_{ij}\cdot\underline{r}_{in}-(\underline{r}_{ij}\cdot\hat{\underline{u}}_{kn})(\underline{r}_{in}\cdot\hat{\underline{u}}_{kn}))\frac{r_{in}^\alpha}{r_{in}^2}\right\} \\
& \left.\left.+(\delta_{\ell n}-\delta_{\ell i})\frac{\underline{r}_{ij}\cdot\hat{\underline{v}}_{kn}}{v_{kn}r_{in}}\left\{r_{ij}^\alpha-(\underline{r}_{ij}\cdot\hat{\underline{v}}_{kn})\hat{v}_{kn}^\alpha-(\underline{r}_{ij}\cdot\underline{r}_{in}-(\underline{r}_{ij}\cdot\hat{\underline{v}}_{kn})(\underline{r}_{in}\cdot\hat{\underline{v}}_{kn}))\frac{r_{in}^\alpha}{r_{in}^2}\right\}\right]\right\}
\end{aligned}
\tag{2.59}
$$

This general formula applies to all atoms $\ell = i, j, k, n$. It must be remembered however, that these formulae apply to just one of the three contributing terms (i.e. one angle ϕ) of the full inversion potential: specifically the inversion angle pertaining to the out-of-plane vector $\underline{r}_{ij}$. The contributions arising from the other vectors $\underline{r}_{ik}$ and $\underline{r}_{in}$ are obtained by the cyclic permutation of the indices in the manner described above. All these force contributions must be added to the final atomic forces.

Formally, the contribution to be added to the atomic virial is given by

$$\mathcal{W} = -\sum_{i=1}^{4}\underline{r}_i\cdot\underline{f}_i \quad . \tag{2.60}$$

However, it is possible to show by thermodynamic arguments (*cf* [28],) or simply from the fact that the sum of forces on atoms j,k and n is equal and opposite to the force on atom i, that the inversion potential makes *no* contribution to the atomic virial.

If the force components f_ℓ^α for atoms $\ell = i, j, k, n$ are calculated using the above formulae, it is easily seen that the contribution to be added to the atomic stress tensor is given by

$$\sigma^{\alpha\beta} = r_{ij}^\alpha f_j^\beta + r_{ik}^\alpha f_k^\beta + r_{in}^\alpha f_n^\beta \quad . \tag{2.61}$$

The sum of the diagonal elements of the stress tensor is zero (since the virial is zero) and the matrix is symmetric.

In DL_POLY_3 inversion forces are handled by the routine INVERSIONS_FORCES.

2.2.8 Tethering Forces

DL_POLY_3 also allows atomic sites to be tethered to a fixed point in space, $\vec{r_0}$, taken as their position at the beginning of the simulation (t = 0). This is also known as position restraining. The specification, which comes as part of the molecular description, requires a tether potential type and the associated interaction parameters.

Note, firstly, that application of tethering potentials means that the momentum will no longer be a conserved quantity of the simulation. Secondly, in constant pressure simulations, where the MD cell changes size or shape, the tethers' reference positions are scaled with the cell vectors.

The tethering potential functions available in DL_POLY_3 are as follows:

1. Harmonic: (**harm**)
$$U(r_{ij}) = \frac{1}{2}k(r_{i0})^2 \tag{2.62}$$
2. Restrained harmonic: (**rhrm**)
$$U(r_{ij}) = \begin{cases} \frac{1}{2}k(r_{i0})^2 & : \quad |r_{i0}| \le r_c \\ \frac{1}{2}kr_c^2 + kr_c(r_{i0} - r_c) & : \quad |r_{i0}| > r_c \end{cases} \tag{2.63}$$
3. Quartic potential: (**quar**)
$$U(r_{ij}) = \frac{k}{2}(r_{i0})^2 + \frac{k'}{3}(r_{i0})^3 + \frac{k''}{4}(r_{i0})^4 \tag{2.64}$$

as in each case r_{io} is the distance between the atom positions at moment $t = t1$ and $t = 0$.

The force on the atom i arising from a tether potential potential is obtained using the general formula:
$$\underline{f}_i = -\frac{1}{r_{i0}}\left[\frac{\partial}{\partial r_{i0}}U(r_{i0})\right]\underline{r}_{i0} \quad . \tag{2.65}$$

The contribution to be added to the atomic virial is given by
$$\mathcal{W} = \underline{r}_{i0} \cdot \underline{f}_i \quad . \tag{2.66}$$

The contribution to be added to the atomic stress tensor is given by
$$\sigma^{\alpha\beta} = -r_{i0}^{\alpha} f_i^{\beta} \quad , \tag{2.67}$$

where α and β indicate the x, y, z components. The atomic stress tensor derived in this way is symmetric.

In DL_POLY_3 tether forces are handled by the routine TETHERS_FORCES.

2.2.9 Frozen Atoms

DL_POLY_3 also allows atoms to be completely immobilized (*i.e.* "frozen" at a fixed point in the MD cell). This is achieved by setting all forces and velocities associated with that atom to zero during each MD timestep. Frozen atoms are signalled by assigning an atom a non-zero value for the freeze parameter in the FIELD file. DL_POLY_3 does not calculate contributions to the virial or the stress tensor arising from the constraints required to freeze atomic positions. Neither does it calculate contributions from intramolecular interactions between frozen atoms.

In DL_POLY_3 the frozen atom option is handled by the subroutine FREEZE_ATOMS.

2.3 The Intermolecular Potential Functions

In this section we outline the two-body, metal, Tersoff, three-body and four-body potential functions in DL_POLY_3 . An important distinction between these and intramolecular (bond) forces in DL_POLY_3 is that they are specified by *atom types* rather than atom indices.

2.3.1 Short Ranged (van der Waals) Potentials

The short ranged pair forces available in DL_POLY_3 are as follows.

1. 12-6 potential: (**12-6**)
$$U(r_{ij}) = \left(\frac{A}{r_{ij}^{12}}\right) - \left(\frac{B}{r_{ij}^{6}}\right) \tag{2.68}$$

2. Lennard-Jones: (**lj**)
$$U(r_{ij}) = 4\epsilon\left[\left(\frac{\sigma}{r_{ij}}\right)^{12} - \left(\frac{\sigma}{r_{ij}}\right)^{6}\right] \tag{2.69}$$

3. n-m potential [31]: (**nm**)
$$U(r_{ij}) = \frac{E_o}{(n-m)}\left[m\left(\frac{r_o}{r_{ij}}\right)^{n} - n\left(\frac{r_o}{r_{ij}}\right)^{m}\right] \tag{2.70}$$

4. Buckingham potential: (**buck**)
$$U(r_{ij}) = A\ \exp\left(-\frac{r_{ij}}{\rho}\right) - \frac{C}{r_{ij}^{6}} \tag{2.71}$$

5. Born-Huggins-Meyer potential: (**bhm**)
$$U(r_{ij}) = A\ \exp[B(\sigma - r_{ij})] - \frac{C}{r_{ij}^{6}} - \frac{D}{r_{ij}^{8}} \tag{2.72}$$

6. Hydrogen-bond (12-10) potential: (**hbnd**)
$$U(r_{ij}) = \left(\frac{A}{r_{ij}^{12}}\right) - \left(\frac{B}{r_{ij}^{10}}\right) \tag{2.73}$$

7. Shifted force n-m potential [31]: (**snm**)
$$\begin{aligned} U(r_{ij}) \;=\; & \frac{\alpha E_o}{(n-m)}\left[m\beta^n\left\{\left(\frac{r_o}{r_{ij}}\right)^{n} - \left(\frac{1}{\gamma}\right)^{n}\right\} - n\beta^m\left\{\left(\frac{r_o}{r_{ij}}\right)^{m} - \left(\frac{1}{\gamma}\right)^{m}\right\}\right] \;+ \\ & \frac{nm\alpha E_o}{(n-m)}\left(\frac{r_{ij}-\gamma r_o}{\gamma r_o}\right)\left\{\left(\frac{\beta}{\gamma}\right)^{n} - \left(\frac{\beta}{\gamma}\right)^{m}\right\} \quad , \end{aligned} \tag{2.74}$$

with

$$\begin{aligned} \alpha &= \frac{(n-m)}{[n\beta^m(1+(m/\gamma-m-1)/\gamma^m) - m\beta^n(1+(n/\gamma-n-1)/\gamma^n)]} \\ \beta &= \gamma\left(\frac{\gamma^{m+1}-1}{\gamma^{n+1}-1}\right)^{\frac{1}{n-m}} \\ \gamma &= \frac{r_{cut}}{r_o} \quad . \end{aligned} \tag{2.75}$$

This peculiar form has the advantage over the standard shifted n-m potential in that both E_o and r_0 (well depth and location of minimum) retain their original values after the shifting process.

8. Morse potential: (**mors**)

$$U(r_{ij}) = E_o \left[\{1 - \exp(-k(r_{ij} - r_o))\}^2 - 1\right] \tag{2.76}$$

9. Tabulation: (**tab**). The potential is defined numerically only.

The parameters defining these potentials are supplied to DL_POLY_3 at run time (see the description of the FIELD file in Section 5.1.3). Each atom type in the system is specified by a unique eight-character label defined by the user. The pair potential is then defined internally by the combination of two atom labels.

As well as the numerical parameters defining the potentials, DL_POLY_3 must also be provided with a cutoff radius r_{vdw}, which sets a range limit on the computation of the interaction. Together with the parameters, the cutoff is used by the subroutine VDW_GENERATE to construct an interpolation array `vvdw` for the potential function over the range 0 to r_{vdw} . A second array `gvdw` is also calculated, which is related to the potential via the formula:

$$G(r_{ij}) = -r_{ij} \frac{\partial}{\partial r_{ij}} U(r_{ij}) \quad , \tag{2.77}$$

and is used in the calculation of the forces. Both arrays are tabulated in units of energy. The use of interpolation arrays, rather than the explicit formulae, makes the routines for calculating the potential energy and atomic forces very general, and enables the use of user defined pair potential functions. DL_POLY_3 also allows the user to read in the interpolation arrays directly from a file (implemented in the VDW_TABLE_READ routine) and the TABLE file (Section 5.1.6). This is particularly useful if the pair potential function has no simple analytical description (e.g. spline potentials).

The force on an atom j derived from one of these potentials is formally calculated with the standard formula:

$$\underline{f}_j = -\frac{1}{r_{ij}} \left[\frac{\partial}{\partial r_{ij}} U(r_{ij}) \right] \underline{r}_{ij} \quad , \tag{2.78}$$

where $\underline{r}_{ij} = \underline{r}_j - \underline{r}_i$. The force on atom i is the negative of this.

The contribution to be added to the atomic virial (for each pair interaction) is

$$\mathcal{W} = -\underline{r}_{ij} \cdot \underline{f}_j \quad . \tag{2.79}$$

The contribution to be added to the atomic stress tensor is given by

$$\sigma^{\alpha\beta} = r_{ij}^{\alpha} f_j^{\beta} \quad , \tag{2.80}$$

where α and β indicate the x, y, z components. The atomic stress tensor derived from the pair forces is symmetric.

Since the calculation of pair potentials assumes a spherical cutoff (r_{vdw}) it is necessary to apply a *long range correction* to the system potential energy and virial. Explicit formulae are needed for each case and are derived as follows. For two atom types a and b, the correction for the potential energy is calculated via the integral

$$U_{corr}^{ab} = 2\pi \frac{N_a N_b}{V} \int_{r_{vdw}}^{\infty} g_{ab}(r) U_{ab}(r) r^2 dr \quad , \tag{2.81}$$

where N_a, N_b are the numbers of atoms of types a and b in the system, V is the system volume and $g_{ab}(r)$ and $U_{ab}(r)$ are the appropriate pair correlation function and pair potential respectively. It is

usual to assume $g_{ab}(r) = 1$ for $r > r_{vdw}$. DL_POLY_3 sometimes makes the additional assumption that the repulsive part of the short ranged potential is negligible beyond r_{vdw} .

The correction for the system virial is

$$\mathcal{W}_{corr}^{ab} = -2\pi \frac{N_a N_b}{V} \int_{r_{vdw}}^{\infty} g_{ab}(r) \frac{\partial}{\partial r} U_{ab}(r) r^3 dr \quad , \tag{2.82}$$

where the same approximations are applied.

Note that these formulae are based on the assumption that the system is reasonably isotropic beyond the cutoff.

In DL_POLY_3 the short ranged forces are calculated by the subroutine VDW_FORCES. The long range corrections are calculated by routine VDW_LRC. The calculation makes use of the Verlet neighbour list (see above).

2.3.2 Metal Potentials

DL_POLY_3 includes density dependent potentials suitable for calculating the properties of metals. The basic model is due to Finnis and Sinclair [32] as implemented by Sutton and Chen [10, 11, 12]. The form of the potential is: (**stch**)

$$U_{sc} = \epsilon \left[\sum_{i<j} \left(\frac{a}{r_{ij}} \right)^n - C \sum_i \rho_i^{1/2} \right] \quad , \tag{2.83}$$

where the *local density*, ρ_i, is given by

$$\rho_i = \sum_j \left(\frac{a}{r_{ij}} \right)^m \quad . \tag{2.84}$$

The Sutton-Chen potential has the advantage that it is decomposable into pair contributions and thus falls within the general tabulation scheme of DL_POLY_3 , where it is treated as a pair interaction, although the metal cutoff (r_{met}) has nothing to do with short ranged one (r_{vdw}). The same form of potential may be used in alloys, through the appropriate choice of parameters ϵ and a. The parameters n and m, however, must be the same for all component elements of the alloy. DL_POLY_3 calculates this potential in two stages: the first calculates the local density, ρ_i, for each atom; and the second calculates the potential energy and forces. Interpolation arrays, `vmet` and `gmet` (METAL_GENERATE) - similar to those in van der Waals interactions 2.3.1, are used in both these stages.

The total force $\underline{f}_j^{tot}$ on an atom j derived from this potential is calculated in the standard way:

$$\underline{f}_j^{tot} = -\underline{\nabla}_j U_{sc} \quad , \tag{2.85}$$

which gives

$$\underline{f}_j^{tot} = -\epsilon \sum_{i \neq j} \left[n \left(\frac{a}{r_{ij}} \right)^n - \frac{Cm}{2} (\rho_j^{-1/2} + \rho_i^{-1/2}) \left(\frac{a}{r_{ij}} \right)^m \right] \left(\frac{1}{r_{ij}^2} \right) \underline{r_{ij}} \quad , \tag{2.86}$$

which is recognizable as a sum of pair forces. For example, the force on atom j due to the atom i:

$$\underline{f}_j = -\epsilon \left[n \left(\frac{a}{r_{ij}} \right)^n - \frac{Cm}{2} (\rho_j^{-1/2} + \rho_i^{-1/2}) \left(\frac{a}{r_{ij}} \right)^m \right] \left(\frac{1}{r_{ij}^2} \right) \underline{r_{ij}} \quad , \tag{2.87}$$

where $\underline{r}_{ij} = \underline{r}_j - \underline{r}_i$. The force on atom i is the negative of this.

With the pair forces thus defined the contribution to be added to the atomic virial *from each atom pair* is then

$$\mathcal{W} = -\underline{r}_{ij} \cdot \underline{f}_j \quad . \tag{2.88}$$

The contribution to be added to the atomic stress tensor is given by

$$\sigma^{\alpha\beta} = r_{ij}^{\alpha} f_j^{\beta} \quad , \tag{2.89}$$

where α and β indicate the x, y, z components. The atomic stress tensor is symmetric.

The long range correction for the system potential is in two parts. Firstly, by analogy with the short ranged potentials, the correction to the local density is obtained by

$$\rho_i = \rho_i^o + 4\pi\bar{\rho}\int_{r_{cut}}^{\infty} \left(\frac{a}{r}\right)^m r^2 dr \quad , \tag{2.90}$$

where ρ_i^o is the uncorrected local density and $\bar{\rho}$ is the *mean particle density*. Evaluating the integral part of the above equation yields

$$\delta\rho = 4\pi\frac{\bar{\rho}a^3}{(m-3)}\left(\frac{a}{r_{cut}}\right)^{m-3} \quad , \tag{2.91}$$

which is the local density correction and is identical for all atoms. The correction is applied immediately after the local density is calculated. The density term of the Sutton-Chen potential needs no further correction. The pair term correction is obtained by analogy with the short ranged potentials and is

$$U_{corr} = 2\pi\frac{N\epsilon\bar{\rho}a^3}{(n-3)}\left(\frac{a}{r_{cut}}\right)^{n-3} \quad . \tag{2.92}$$

The correction to the local density having already been applied.

To estimate the virial correction we assume the corrected local densities are constants (i.e. independent of distance - at least beyond the range r_{met}). This allows the virial correction to be computed by the methods used in the short ranged potentials. The result is:

$$\mathcal{W}_{corr} = -2\pi\bar{\rho}a^3\left\{\frac{nN\epsilon}{(n-3)}\left(\frac{a}{r_{cut}}\right)^{n-3} - \frac{m\epsilon^2C^2}{(m-3)}\left(\frac{a}{r_{cut}}\right)^{m-3}\sum_i^N \rho_i^{-1/2}\right\} \quad . \tag{2.93}$$

This correction may be used as it stands, or with the further approximation:

$$\sum_i^N \rho_i^{-1/2} = \frac{N}{<\rho_i^{1/2}>} \quad , \tag{2.94}$$

where $<\rho_i^{1/2}>$ is regarded as a constant of the system.

In DL_POLY_3 the metal forces are handled by the routine METAL_FORCES. The local density is calculated by routine METAL_LD_COMPUTE. The long range corrections are calculated by METAL_LRC.

2.3.3 Tersoff Potential

The Tersoff [13] potential has been developed to be used in multi-component covalent systems. It has 11 atomic and 2 bi-atomic parameters. The energy is modelled as a sum of pair-like interactions,

where, however the coefficient of the attractive term in the pairlike potential (which plays the role of a bond order) depends on the local environment giving a many-body potential.

The form of the Tersoff potential is: (**ters**)

$$U_{ij} = f_C(r_{ij})\ [f_R(r_{ij}) - \gamma_{ij}\ f_A(r_{ij})], \tag{2.95}$$

where

$$f_R(r_{ij}) = A_{ij}\ \exp(-a_{ij}\ r_{ij})\ ,\quad f_A(r_{ij}) = B_{ij}\ \exp(-b_{ij}\ r_{ij}) \tag{2.96}$$

$$f_C(r_{ij}) = \begin{cases} 1 & : \quad r_{ij} < R_{ij} \\ \frac{1}{2} + \frac{1}{2}\cos[\pi\ (r_{ij} - R_{ij})/(r_{ij} - R_{ij})] & : \quad R_{ij} < r_{ij} < S_{ij} \\ 0 & : \quad r_{ij} > S_{ij} \end{cases} \tag{2.97}$$

$$\gamma_{ij} = \chi_{ij}\ (1 + \beta_i^{\eta_i}\ \mathcal{L}_{ij}^{\eta_i})^{-1/2\eta_i}\ ,\quad \mathcal{L}_{ij} = \sum_{k\neq i,j} f_C(r_{ik})\ \omega_{ik}\ g(\theta_{ijk})$$
$$g(\theta_{ijk}) = 1 + c_i^2/d_i^2 - c_i^2/[d_i^2 + (h_i - \cos\theta_{ijk})^2] \tag{2.98}$$

with further mixed parameters defined as

$$\begin{aligned} a_{ij} &= (a_i + a_j)/2 \quad , \quad b_{ij} = (b_i + b_j)/2 \\ A_{ij} &= (A_i A_j)^{1/2} \quad , \quad B_{ij} = (B_i B_j)^{1/2} \\ R_{ij} &= (R_i R_j)^{1/2} \quad , \quad S_{ij} = (S_i S_j)^{1/2} \quad . \end{aligned} \tag{2.99}$$

Here $i,\ j$ and k label the atoms in the system, r_{ij} is the length of ij bond, and θ_{ijk} is the bond angle between bonds ij and ik. Singly subscripted parameters (11), such as a_i and η_i, depend only on the type of atom.

The chemistry between different atom types is locked in the two sets bi-atomic parameters χ_{ij} and ω_{ij}:

$$\begin{aligned} \chi_{ii} &= 1 \quad , \quad \chi_{ij} = \chi_{ji} \\ \omega_{ii} &= 1 \quad , \quad \omega_{ij} = \omega_{ji} \quad , \end{aligned} \tag{2.100}$$

which define only one independent parameter each per pair of atom types. The χ parameter is used to strengthen or weaken the heteropolar bonds, relative to the value obtained by simple interpolation. The ω parameter is used to permit greater flexibility when dealing with more drastically different types of atoms.

The force on an atom ℓ derived from this potential is formally calculated with the formula:

$$f_\ell^\alpha = -\frac{\partial}{\partial r_\ell^\alpha} E_{\texttt{tersoff}} = \frac{1}{2}\sum_{i\neq j} -\frac{\partial}{\partial r_\ell^\alpha} U_{ij} \quad , \tag{2.101}$$

with atomic label ℓ being one of i, j, k and α indicating the x, y, z component. The derivative after the summation is worked out as

$$-\frac{\partial U_{ij}}{\partial r_\ell^\alpha} = -\frac{\partial}{\partial r_\ell^\alpha} f_C(r_{ij}) f_R(r_{ij}) + \gamma_{ij}\frac{\partial}{\partial r_\ell^\alpha} f_C(r_{ij}) f_A(r_{ij}) + f_C(r_{ij}) f_A(r_{ij})\frac{\partial}{\partial r_\ell^\alpha}\gamma_{ij} \quad , \tag{2.102}$$

with the contributions from the first in the forms:

$$\begin{aligned} -\frac{\partial}{\partial r_\ell^\alpha} f_C(r_{ij}) f_R(r_{ij}) \quad &= \quad -\left\{ f_C(r_{ij})\frac{\partial}{\partial r_{ij}} f_R(r_{ij}) + f_R(r_{ij})\frac{\partial}{\partial r_{ij}} f_C(r_{ij}) \right\} \times \\ & \qquad \left\{ \delta_{j\ell}\frac{r_{i\ell}^\alpha}{r_{i\ell}} - \delta_{i\ell}\frac{r_{\ell j}^\alpha}{r_{\ell j}} \right\} \end{aligned} \tag{2.103}$$

$$\gamma_{ij}\frac{\partial}{\partial r_\ell^\alpha}f_C(r_{ij})f_A(r_{ij}) = \gamma_{ij}\left\{f_C(r_{ij})\frac{\partial}{\partial r_{ij}}f_A(r_{ij}) + f_A(r_{ij})\frac{\partial}{\partial r_{ij}}f_C(r_{ij})\right\}\times \left\{\delta_{j\ell}\frac{r_{i\ell}^\alpha}{r_{i\ell}} - \delta_{i\ell}\frac{r_{\ell j}^\alpha}{r_{\ell j}}\right\} \quad , \tag{2.104}$$

and from the third (angular) term:

$$f_C(r_{ij})f_A(r_{ij})\frac{\partial}{\partial r_\ell^\alpha}\gamma_{ij} = f_C(r_{ij})f_A(r_{ij})\ \chi_{ij} \quad \times \left(-\frac{1}{2}\right)\left(1+\beta_i{}^{\eta_i}\ \mathcal{L}_{ij}^{\eta_i}\right)^{-\frac{1}{2\eta_i}-1}\beta_i{}^{\eta_i}\ \mathcal{L}_{ij}^{\eta_i-1}\frac{\partial}{\partial r_\ell^\alpha}\mathcal{L}_{ij} \quad , \tag{2.105}$$

where

$$\frac{\partial}{\partial r_\ell^\alpha}\mathcal{L}_{ij} = \frac{\partial}{\partial r_\ell^\alpha}\sum_{k\neq i,j}\omega_{ik}\ f_C(r_{ik})\ g(\theta_{ijk}) \quad . \tag{2.106}$$

The angular term can have three different contributions depending on the index of the particle participating in the interaction:

$$\ell = i \quad : \quad \frac{\partial}{\partial r_i^\alpha}\mathcal{L}_{ij} = \sum_{k\neq i,j}\omega_{ik}\left[g(\theta_{ijk})\frac{\partial}{\partial r_i^\alpha}f_C(r_{ik}) + f_C(r_{ik})\frac{\partial}{\partial r_i^\alpha}g(\theta_{ijk})\right] \tag{2.107}$$

$$\ell = j \quad : \quad \frac{\partial}{\partial r_j^\alpha}\mathcal{L}_{ij} = \sum_{k\neq i,j}\omega_{ik}\ f_C(r_{ik})\frac{\partial}{\partial r_j^\alpha}g(\theta_{ijk}) \tag{2.108}$$

$$\ell \neq i,j \quad : \quad \frac{\partial}{\partial r_\ell^\alpha}\mathcal{L}_{ij} = \omega_{i\ell}\left[g(\theta_{ij\ell})\frac{\partial}{\partial r_\ell^\alpha}f_C(r_{i\ell}) + f_C(r_{i\ell})\frac{\partial}{\partial r_\ell^\alpha}g(\theta_{ij\ell})\right] \quad . \tag{2.109}$$

The derivative of $g(\theta_{ijk})$ is worked out in the following manner:

$$\frac{\partial}{\partial r_\ell^\alpha}g(\theta_{ijk}) = \frac{\partial g(\theta_{ijk})}{\partial\theta_{ijk}}\ \frac{-1}{\sin\theta_{ijk}}\ \frac{\partial}{\partial r_\ell^\alpha}\left\{\frac{\underline{r}_{ij}\cdot\underline{r}_{ik}}{r_{ij}\ r_{ik}}\right\} \quad , \tag{2.110}$$

where

$$\frac{\partial g(\theta_{ijk})}{\partial\theta_{ijk}} = \frac{2\ c_i^2(h_i-\cos\theta_{ijk})\ \sin\theta_{ijk}}{[d_i^2+(h_i-\cos\theta_{ijk})^2]^2} \tag{2.111}$$

$$\frac{\partial}{\partial r_\ell^\alpha}\left\{\frac{\underline{r}_{ij}\cdot\underline{r}_{ik}}{r_{ij}r_{ik}}\right\} = (\delta_{\ell j}-\delta_{\ell i})\frac{r_{ik}^\alpha}{r_{ij}r_{ik}} + (\delta_{\ell k}-\delta_{\ell i})\frac{r_{ij}^\alpha}{r_{ij}r_{ik}} - \cos(\theta_{jik})\left\{(\delta_{\ell j}-\delta_{\ell i})\frac{r_{ij}^\alpha}{r_{ij}^2} + (\delta_{\ell k}-\delta_{\ell i})\frac{r_{ik}^\alpha}{r_{ik}^2}\right\} \quad . \tag{2.112}$$

The contribution to be added to the atomic virial can be derived as

$$\mathcal{W} = 3V\frac{\partial E_{\texttt{tersoff}}}{\partial V} = \frac{3\ V}{2}\sum_{i\neq j}\frac{\partial U_{ij}}{\partial V} \tag{2.113}$$

$$\mathcal{W} = \frac{1}{2}\sum_{i\neq j}\left[\frac{\partial}{\partial r_{ij}}f_C(r_{ij})f_R(r_{ij}) - \gamma_{ij}\frac{\partial}{\partial r_{ij}}f_C(r_{ij})f_A(r_{ij})\right]r_{ij} - \left(-\frac{1}{2}\right)f_C(r_{ij})f_A(r_{ij})\ \chi_{ij}\left(1+\beta_i{}^{\eta_i}\ \mathcal{L}_{ij}^{\eta_i}\right)^{-\frac{1}{2\eta_i}-1}\beta_i{}^{\eta_i}\ \mathcal{L}_{ij}^{\eta_i-1}\times \sum_{k\neq i,j}\omega_{ik}\ g(\theta_{ijk})\left[\frac{\partial}{\partial r_{ik}}f_C(r_{ik})\right]r_{ik} \quad . \tag{2.114}$$

The contribution to be added to the atomic stress tensor is given by

$$\sigma^{\alpha\beta} = -r_i^{\alpha} f_i^{\beta} \quad , \tag{2.115}$$

where α and β indicate the x, y, z components. The stress tensor is symmetric.

Interpolation arrays, `vter` and `gter` (set up in TERSOFF_GENERATE) - similar to those in van der Waals interactions 2.3.1, are used in the calculation of the Tersoff forces, virial and stress.

The Tersoff potentials are very short ranged, typically of order 3 Å. This property, plus the fact that Tersoff potentials (two- and three-body contributions) scale as N^3, where N is the number of particles, makes it essential that these terms are calculated by the link-cell method [33].

DL_POLY_3 applies no long range corrections to the Tersoff potentials. In DL_POLY_3 Tersoff forces are handled by the routine TERSOFF_FORCES.

2.3.4 Three Body Potentials

The three-body potentials in DL_POLY_3 are mostly valence angle forms. (They are primarily included to permit simulation of amorphous materials e.g. silicate glasses.) However, these have been extended to include the Dreiding [15] hydrogen bond. The potential forms available are as follows:

1. Harmonic: (**harm**)
$$U(\theta_{jik}) = \frac{k}{2}(\theta_{jik} - \theta_0)^2 \tag{2.116}$$

2. Truncated harmonic: (**thrm**)
$$U(\theta_{jik}) = \frac{k}{2}(\theta_{jik} - \theta_0)^2 \exp[-(r_{ij}^8 + r_{ik}^8)/\rho^8] \tag{2.117}$$

3. Screened Harmonic: (**shrm**)
$$U(\theta_{jik}) = \frac{k}{2}(\theta_{jik} - \theta_0)^2 \exp[-(r_{ij}/\rho_1 + r_{ik}/\rho_2)] \tag{2.118}$$

4. Screened Vessal [25]: (**bvs1**)
$$\begin{aligned} U(\theta_{jik}) \quad = \quad & \frac{k}{8(\theta_{jik} - \pi)^2} \left\{ \left[(\theta_0 - \pi)^2 - (\theta_{jik} - \pi)^2 \right]^2 \right\} \times \\ & \exp[-(r_{ij}/\rho_1 + r_{ik}/\rho_2)] \end{aligned} \tag{2.119}$$

5. Truncated Vessal [26]: (**bvs2**)
$$\begin{aligned} U(\theta_{jik}) \quad = \quad & k \, [\theta_{jik}^a (\theta_{jik} - \theta_0)^2 (\theta_{jik} + \theta_0 - 2\pi)^2 - \\ & \frac{a}{2}\pi^{a-1}(\theta_{jik} - \theta_0)^2(\pi - \theta_0)^3] \; \exp[-(r_{ij}^8 + r_{ik}^8)/\rho^8] \end{aligned} \tag{2.120}$$

6. Dreiding hydrogen bond [15]: (**hbnd**)
$$U(\theta_{jik}) = D_{hb} \; \cos^4(\theta_{jik}) \; [5(R_{hb}/r_{jk})^{12} - 6(R_{hb}/r_{jk})^{10}] \tag{2.121}$$

Note that for the hydrogen bond, the hydrogen atom *must* be the central atom. Several of these functions are identical to those appearing in the *intra*-molecular valence angle descriptions above. There are significant differences in implementation however, arising from the fact that the three-body potentials are regarded as *inter*-molecular. Firstly, the atoms involved are defined by atom types, not specific indices. Secondly, there are *no* excluded atoms arising from the three-body terms. (The inclusion of other potentials, for example pair potentials, may in fact be essential to maintain the structure of the system.)

The three-body potentials are very short ranged, typically of order 3 Å. This property, plus the fact that three-body potentials scale as N^4, where N is the number of particles, makes it essential that these terms are calculated by the link-cell method [33].

The calculation of the forces, virial and stress tensor as described in the section valence angle potentials above.

DL_POLY_3 applies no long range corrections to the three-body potentials. The three-body forces are calculated by the routine THREE_BODY_FORCES.

2.3.5 Four Body Potentials

The four-body potentials in DL_POLY_3 are entirely inversion angle forms, primarily included to permit simulation of amorphous materials (particularly borate glasses). The potential forms available in DL_POLY_3 are as follows:

1. Harmonic: (**harm**)
$$U(\phi_{ijkn}) = \frac{k}{2}\,(\phi_{ijkn} - \phi_0)^2 \tag{2.122}$$

2. Harmonic cosine: (**hcos**)
$$U(\phi_{ijkn}) = \frac{k}{2}\,(\cos(\phi_{ijkn}) - \cos(\phi_0))^2 \tag{2.123}$$

3. Planar potential: (**plan**)
$$U(\phi_{ijkn}) = A\,[1 - \cos(\phi_{ijkn})] \tag{2.124}$$

These functions are identical to those appearing in the *intra*-molecular inversion angle descriptions above. There are significant differences in implementation however, arising from the fact that the four-body potentials are regarded as *inter*-molecular. Firstly, the atoms involved are defined by atom types, not specific indices. Secondly, there are *no* excluded atoms arising from the four-body terms. (The inclusion of other potentials, for example pair potentials, may in fact be essential to maintain the structure of the system.)

The four-body potentials are very short ranged, typically of order 3 Å. This property, plus the fact that four-body potentials scale as N^4, where N is the number of particles, makes it essential that these terms are calculated by the link-cell method [33].

The calculation of the forces, virial and stress tensor described in the section on inversion angle potentials above.

DL_POLY_3 applies no long range corrections to the four body potentials. The four-body forces are calculated by the routine FOUR_BODY_FORCES.

2.4 Long Ranged Electrostatic (coulombic) Potentials

DL_POLY_3 incorporates several techniques for dealing with long ranged electrostatic potentials[2]. These are as follows:

1. Direct Coulomb sum
2. Force shifted Coulomb sum
3. Coulomb sum with distance dependent dielectric
4. Reaction field
5. Smoothed Particle Mesh Ewald (SPME)

All of these can be used in conjunction with the shell model technique used to account for ions polarisation.

The SPME technique is restricted to periodic systems only. (Users must exercise care when using pseudo-periodic boundary conditions.) The other techniques can be used with either periodic or non-periodic systems safely, although in the case of the direct Coulomb sum there are likely to be problems with convergence.

DL_POLY_3 will correctly handle the electrostatics of both molecular and atomic species. However, it is assumed that the system is electrically neutral. A warning message is printed if the system is found to be charged, but otherwise the simulation proceeds as normal.

Note that DL_POLY_3 does not use the basic Ewald method, which is an option in DL_POLY_2, on account of it being too slow for large scale systems. The SPME method is the standard Ewald method in DL_POLY_3 .

2.4.1 Direct Coulomb Sum

Use of the direct Coulomb sum is sometimes necessary for accurate simulation of isolated (non-periodic) systems. It is *not* recommended for periodic systems.

The interaction potential for two charged ions is

$$U(r_{ij}) = \frac{1}{4\pi\epsilon_0}\frac{q_i q_j}{r_{ij}} \quad , \tag{2.125}$$

with q_ℓ the charge on an atom labelled ℓ, and r_{ij} the magnitude of the separation vector $\underline{r}_{ij} = \underline{r}_j - \underline{r}_i$.

The force on an atom j derived from this force is

$$\underline{f}_j = \frac{1}{4\pi\epsilon_0}\frac{q_i q_j}{r_{ij}^3}\underline{r}_{ij} \quad , \tag{2.126}$$

with the force on atom i the negative of this.

The contribution to the atomic virial is

$$\mathcal{W} = -\frac{1}{4\pi\epsilon_0}\frac{q_i q_j}{r_{ij}} \quad , \tag{2.127}$$

[2]Unlike the other elements of the force field, the electrostatic forces are NOT specified in the input FIELD file, but by setting appropriate directives in the CONTROL file. See Section 5.1.1.

which is simply the negative of the potential term.

The contribution to be added to the atomic stress tensor is

$$\sigma^{\alpha\beta} = r_{ij}^{\alpha} f_j^{\beta} \quad , \tag{2.128}$$

where α, β are x, y, z components. The atomic stress tensor is symmetric.

In DL_POLY_3 these forces are handled by the subroutine COUL_CP_FORCES.

2.4.2 Force Shifted Coulomb Sum

This form of the Coulomb sum has the advantage that it drastically reduces the range of electrostatic interactions, without giving rise to a violent step in the potential energy at the cutoff. Its main use is for preliminary preparation of systems and it is not recommended for realistic models.

The form of the simple truncated and shifted potential function is

$$U(r_{ij}) = \frac{q_i q_j}{4\pi\epsilon_0} \left\{ \frac{1}{r_{ij}} - \frac{1}{r_{cut}} \right\} \quad , \tag{2.129}$$

with q_ℓ the charge on an atom labelled ℓ, r_{cut} the cutoff radius and r_{ij} the magnitude of the separation vector $\underline{r}_{ij} = \underline{r}_j - \underline{r}_i$.

A further refinement of this approach is to truncate the $1/r$ potential at $r_{\rm cut}$ and add a linear term to the potential in order to make both the energy and the force zero at the cutoff. This removes the heating effects that arise from the discontinuity in the forces at the cutoff in the simple truncated and shifted potential (the formula above). (The physics of this potential, however, is little better. It is only recommended for very crude structure optimizations.)

The Force Shifted potential is thus

$$U(r_{ij}) = \frac{q_i q_j}{4\pi\epsilon_0} \left[\frac{1}{r_{ij}} + \frac{r_{ij}}{r_{\rm cut}^2} - \frac{2}{r_{\rm cut}} \right] \quad , \tag{2.130}$$

with the force on an atom j given by

$$\underline{f}_j = \frac{q_i q_j}{4\pi\epsilon_0} \left[\frac{1}{r_{ij}^3} - \frac{1}{r_{ij} r_{\rm cut}^2} \right] \underline{r}_{ij} \quad , \tag{2.131}$$

with the force on atom i the negative of this.

The contribution to the atomic virial is

$$\mathcal{W} = -\underline{r}_{ij} \cdot \underline{f}_j \quad , \tag{2.132}$$

which is *not* the negative of the potential term in this case.

The contribution to be added to the atomic stress tensor is given by

$$\sigma^{\alpha\beta} = r_{ij}^{\alpha} f_j^{\beta} \quad , \tag{2.133}$$

where α, β are x, y, z components. The atomic stress tensor is symmetric.

In DL_POLY_3 these forces are handled by the routine COUL_FSCP_FORCES.

2.4.3 Coulomb Sum with Distance Dependent Dielectric

This potential attempts to address the difficulties of applying the direct Coulomb sum, without the brutal truncation of the previous case. It hinges on the assumption that the electrostatic forces are effectively 'screened' in real systems - an effect which is approximated by introducing a dielectric term that increases with distance.

The interatomic potential for two charged ions is

$$U(r_{ij}) = \frac{1}{4\pi\epsilon_0\epsilon(r_{ij})}\frac{q_i q_j}{r_{ij}} \quad , \tag{2.134}$$

with q_ℓ the charge on an atom labelled ℓ, and r_{ij} the magnitude of the separation vector $\underline{r}_{ij} = \underline{r}_j - \underline{r}_i$. $\epsilon(r)$ is the distance dependent dielectric function. In DL_POLY_3 it is assumed that this function has the form

$$\epsilon(r) \;=\; \epsilon\, r \quad , \tag{2.135}$$

where ϵ is a constant. Inclusion of this term effectively accelerates the rate of convergence of the Coulomb sum.

The force on an atom j derived from this potential is

$$\underline{f}_j = \frac{1}{2\pi\epsilon_0\epsilon}\frac{q_i q_j}{r_{ij}^4}\underline{r}_{ij} \quad , \tag{2.136}$$

with the force on atom i the negative of this.

The contribution to the atomic virial is

$$\mathcal{W} = -\underline{r}_{ij}\cdot\underline{f}_j \quad , \tag{2.137}$$

which is -2 times the potential term.

The contribution to be added to the atomic stress tensor is given by

$$\sigma^{\alpha\beta} = r_{ij}^{\alpha} f_j^{\beta} \quad , \tag{2.138}$$

where α, β are x, y, z components. The atomic stress tensor is symmetric.

In DL_POLY_3 these forces are handled by the routine COUL_DDDP_FORCES.

2.4.4 Reaction Field

In the reaction field method it is assumed that any given molecule is surrounded by a spherical cavity of finite radius within which the electrostatic interactions are calculated explicitly. Outside the cavity the system is treated as a dielectric continuum. The occurrence of any net dipole within the cavity induces a polarisation in the dielectric, which in turn interacts with the given molecule. The model allows the replacement of the infinite Coulomb sum by a finite sum plus the reaction field.

The reaction field model coded into DL_POLY_3 is the implementation of Neumann based on charge-charge interactions [34]. In this model, the total coulombic potential is given by

$$U_c = \frac{1}{4\pi\epsilon_0}\sum_{j<n} q_j q_n \left[\frac{1}{r_{nj}} + \frac{B_0 r_{nj}^2}{2R_c^3}\right] \quad , \tag{2.139}$$

where the second term on the right is the reaction field correction to the explicit sum, with R_c the radius of the cavity. The constant B_0 is defined as

$$B_0 = \frac{2(\epsilon_1 - 1)}{(2\epsilon_1 + 1)} \quad , \tag{2.140}$$

with ϵ_1 the dielectric constant outside the cavity. The effective pair potential is therefore

$$U(r_{ij}) = \frac{1}{4\pi\epsilon_0} q_i q_j \left[\frac{1}{r_{ij}} + \frac{B_0 r_{ij}^2}{2R_c^3} \right] \quad . \tag{2.141}$$

This expression unfortunately leads to large fluctuations in the system coulombic energy, due to the large 'step' in the function at the cavity boundary. In DL_POLY_3 this is countered by subtracting the value of the potential at the cavity boundary from each pair contribution. The term subtracted is

$$\frac{1}{4\pi\epsilon_0} \frac{q_i q_j}{R_c} \left[1 + \frac{B_0}{2} \right] \quad . \tag{2.142}$$

The effective pair force on an atom j arising from another atom n within the cavity is given by

$$\underline{f}_j = \frac{q_i q_j}{4\pi\epsilon_0} \left[\frac{1}{r_{ij}^3} - \frac{B_0}{R_c^3} \right] \underline{r}_{ij} \quad . \tag{2.143}$$

The contribution of each effective pair interaction to the atomic virial is

$$\mathcal{W} = -\underline{r}_{ij} \cdot \underline{f}_j \tag{2.144}$$

and the contribution to the atomic stress tensor is

$$\sigma^{\alpha\beta} = r_{ij}^{\alpha} f_j^{\beta} \quad , \tag{2.145}$$

where α, β are x, y, z components. The atomic stress tensor is symmetric.

In DL_POLY_3 the reaction field is handled by the subroutine COUL_RFP_FORCES.

2.4.5 Smoothed Particle Mesh Ewald

The Ewald sum [18] is the best technique for calculating electrostatic interactions in a periodic (or pseudo-periodic) system.

The basic model for a neutral periodic system is a system of charged point ions mutually interacting via the Coulomb potential. The Ewald method makes two amendments to this simple model. Firstly each ion is effectively neutralised (at long range) by the superposition of a spherical gaussian cloud of opposite charge centred on the ion. The combined assembly of point ions and gaussian charges becomes the *Real Space* part of the Ewald sum, which is now short ranged and treatable by the methods described above (Section 2)[3]. The second modification is to superimpose a second set of gaussian charges, this time with the same charges as the original point ions and again centred on the point ions (so nullifying the effect of the first set of gaussians). The potential due to these gaussians is obtained from Poisson's equation and is solved as a Fourier series in *Reciprocal Space*. The complete Ewald sum requires an additional correction, known as the self energy correction,

[3]Strictly speaking, the real space sum ranges over all periodic images of the simulation cell, but in the DL_POLY_3 implementation, the parameters are chosen to restrict the sum to the simulation cell and its nearest neighbours i.e. the *minimum images* of the cell contents.

which arises from a gaussian acting on its own site, and is constant. Ewald's method, therefore, replaces a potentially infinite sum in real space by two finite sums: one in real space and one in reciprocal space; and the self energy correction.

For molecular systems, as opposed to systems comprised simply of point ions, additional modifications EWALD_EXCL_FORCES are necessary to correct for the excluded (intra-molecular) coulombic interactions. In the real space sum these are simply omitted. In reciprocal space however, the effects of individual gaussian charges cannot easily be extracted, and the correction is made in real space. It amounts to removing terms corresponding to the potential energy of an ion ℓ due to the gaussian charge on a neighbouring charge m (or *vice versa*). This correction appears as the final term in the full Ewald formula below. The distinction between the error function erf and the more usual complementary error function $erfc$ found in the real space sum, should be noted.

The same considerations and modifications EWALD_FROZEN_FORCES are taken into account for frozen atoms, which mutual coulombic interaction must be excluded.

The total electrostatic energy is given by the following formula.

$$\begin{aligned} U_c \;=\; & \frac{1}{2V_o\epsilon_0}\sum_{\underline{k}\neq\ vek0}^{\infty}\frac{\exp(-k^2/4\alpha^2)}{k^2}|\sum_{j}^{N}q_j\exp(-i\underline{k}\cdot\underline{r}_j)|^2+\frac{1}{4\pi\epsilon_0}\sum_{n<j}^{N^*}\frac{q_jq_n}{r_{nj}}erfc(\alpha r_{nj})- \\ & \frac{1}{4\pi\epsilon_0}\sum_{molecules}\sum_{\ell\leq m}^{M^*}q_\ell q_m\left\{\delta_{\ell m}\frac{\alpha}{\sqrt{\pi}}+\frac{erf(\alpha r_{\ell m})}{r_{\ell m}^{1-\delta_{\ell m}}}\right\}- \\ & \frac{1}{4\pi\epsilon_0}\sum_{\ell\leq m}^{F^*}q_\ell q_m\left\{\delta_{\ell m}\frac{\alpha}{\sqrt{\pi}}+\frac{erf(\alpha r_{\ell m})}{r_{\ell m}^{1-\delta_{\ell m}}}\right\}-\frac{1}{4\pi\epsilon_0}\frac{1}{V_o\alpha^2}\left\{\sum_{j}^{N}q_j\right\}^2 \quad , \end{aligned} \tag{2.146}$$

where N is the number of ions in the system and N^* the same number discounting any excluded (intramolecular and frozen) interactions. M^* represents the number of excluded atoms in a given molecule. F^* represents the number of frozen atoms in the MD cell. V_o is the simulation cell volume and $\underline{k}$ is a reciprocal lattice vector defined by

$$\underline{k}=\ell\underline{u}+m\underline{v}+n\underline{w} \quad , \tag{2.147}$$

where ℓ, m, n are integers and $\underline{u}, \underline{v}, \underline{w}$ are the *reciprocal space* basis vectors. Both V_o and $\underline{u}, \underline{v}, \underline{w}$ are derived from the vectors $(\underline{a}, \underline{b}, \underline{c})$ defining the simulation cell. Thus

$$V_o=|\underline{a}\cdot\underline{b}\times\underline{c}| \tag{2.148}$$

and

$$\begin{aligned} \underline{u} &= 2\pi\frac{\underline{b}\times\underline{c}}{\underline{a}\cdot\underline{b}\times\underline{c}} \\ \underline{v} &= 2\pi\frac{\underline{c}\times\underline{a}}{\underline{a}\cdot\underline{b}\times\underline{c}} \\ \underline{w} &= 2\pi\frac{\underline{a}\times\underline{b}}{\underline{a}\cdot\underline{b}\times\underline{c}} \quad . \end{aligned} \tag{2.149}$$

With these definitions, the Ewald formula above is applicable to general periodic systems. The last term in the Ewald formula above is the Fuchs correction [35] for electrically non-neutral MD cells which prevents the build-up of a charged background and the introduction of extra pressure due to it.

In practice the convergence of the Ewald sum is controlled by three variables: the real space cutoff r_{cut}; the convergence parameter α and the largest reciprocal space vector $\underline{k}_{max}$ used in the reciprocal

space sum. These are discussed more fully in Section 4.3.5. DL_POLY_3 can provide estimates if requested (see CONTROL file description 5.1.1.

As its name implies the Smoothed Particle Mesh Ewald (SPME) method is a modification of the standard Ewald method. DL_POLY_3 implements the SPME method of Essmann *et al.* [36]. Formally, this method is capable of treating van der Waals forces also, but in DL_POLY_3 it is confined to electrostatic forces only. The main difference from the standard Ewald method is in its treatment of the the reciprocal space terms. By means of an interpolation procedure involving (complex) B-splines, the sum in reciprocal space is represented on a three dimensional rectangular grid. In this form the Fast Fourier Transform (FFT) may be used to perform the primary mathematical operation, which is a 3D convolution. The efficiency of these procedures greatly reduces the cost of the reciprocal space sum when the range of $\underline{k}$ vectors is large. The method (briefly) is as follows (for full details see [36]):

1. Interpolation of the $exp(-i\ \underline{k}\cdot\underline{r}_j)$ terms (given here for one dimension):

$$exp(2\pi i\ u_j k/L) \approx b(k) \sum_{\ell=-\infty}^{\infty} M_n(u_j-\ell)\ exp(2\pi i\ k\ell/K) \quad , \tag{2.150}$$

in which k is the integer index of the $\underline{k}$ vector in a principal direction, K is the total number of grid points in the same direction and u_j is the fractional coordinate of ion j scaled by a factor K (i.e. $u_j = Ks_j^x$) . Note that the definition of the B-splines implies a dependence on the integer K, which limits the formally infinite sum over ℓ. The coefficients $M_n(u)$ are B-splines of order n and the factor $b(k)$ is a constant computable from the formula:

$$b(k) = exp(2\pi i\ (n-1)k/K) \left[\sum_{\ell=0}^{n-2} M_n(\ell+1)\ exp(2\pi i\ k\ell/K)\right]^{-1} \quad . \tag{2.151}$$

2. Approximation of the structure factor $S(\underline{k})$:

$$S(\underline{k}) \approx b_1(k_1)\ b_2(k_2)\ b_3(k_3)\ Q^\dagger(k_1,k_2,k_3) \quad , \tag{2.152}$$

where $Q^\dagger(k_1,k_2,k_3)$ is the discrete Fourier transform of the *charge array* $Q(\ell_1,\ell_2,\ell_3)$ defined as

$$\begin{aligned} Q(\ell_1,\ell_2,\ell_3) \quad = \quad & \sum_{j=1}^{N} q_j \sum_{n_1,n_2,n_3} M_n(u_{1j}-\ell_1-n_1L_1)\ \times\ M_n(u_{2j}-\ell_2-n_2L_2)\ \times \\ & M_n(u_{3j}-\ell_3-n_3L_3) \quad , \end{aligned} \tag{2.153}$$

in which the sums over $n_{1,2,3}$ etc are required to capture contributions from all relevant periodic cell images (which in practice means the nearest images).

3. Approximating the reciprocal space energy U_{recip}:

$$U_{recip} = \frac{1}{2V_o\epsilon_0} \sum_{k_1,k_2,k_3} G^\dagger(k_1,k_2,k_3)\ Q(k_1,k_2,k_3) \quad , \tag{2.154}$$

where $G^\dagger$ is the discrete Fourier transform of the function

$$G(k_1,k_2,k_3) = \frac{\exp(-k^2/4\alpha^2)}{k^2}\ B(k_1,k_2,k_3)\ (Q^\dagger(k_1,k_2,k_3))^* \quad , \tag{2.155}$$

in which $(Q^{\dagger}(k_1,k_2,k_3))^*$ is the complex conjugate of $Q^{\dagger}(k_1,k_2,k_3)$ and

$$B(k_1,k_2,k_3) = |b_1(k_1)|^2 \ |b_2(k_2)|^2 \ |b_3(k_3)|^2 \quad . \tag{2.156}$$

The function $G(k_1,k_2,k_3)$ is thus a relatively simple product of the gaussian screening term appearing in the conventional Ewald sum, the function $B(k_1,k_2,k_3)$ and the discrete Fourier transform of $Q(k_1,k_2,k_3)$.

4. Calculating the atomic forces, which are given formally by:

$$f_j^{\alpha} = -\frac{\partial U_{recip}}{\partial r_j^{\alpha}} = -\frac{1}{V_o \epsilon_0} \sum_{k_1,k_2,k_3} G^{\dagger}(k_1,k_2,k_3) \ \frac{\partial Q(k_1,k_2,k_3)}{\partial r_j^{\alpha}} \quad . \tag{2.157}$$

Fortunately, due to the recursive properties of the B-splines, these formulae are easily evaluated.

The virial and the stress tensor are calculated in the same manner as for the conventional Ewald sum.

The DL_POLY_3 subroutines required to calculate the SPME contributions are:

1. SPME_CONTAINER containing
   (a) BSPGEN, which calculates the B-splines
   (b) BSPCOE, which calculates B-spline coefficients
   (c) SPL_CEXP, which calculates the FFT and B-spline complex exponentials
2. PARALLEL_FFT and GPFA_WRAP (native DL_POLY_3 subroutines that respect the domain decomposition concept) which calculate the 3D complex fast Fourier transforms
3. EWALD_SPME_FORCES, which calculates the reciprocal space contributions (uncorrected)
4. EWALD_REAL_FORCES, which calculates the real space contributions (corrected)
5. EWALD_EXCL_FORCES, which calculates the reciprocal space corrections due to the coulombic exclusions in intramolecular interactions
6. EWALD_FROZEN_FORCES, which calculates the reciprocal space corrections due to the exclusion interactions between frozen atoms
7. TWO_BODY_FORCES, in which all of the above subroutines are called sequentially and also the Fuchs correction [35] for electrically non-neutral MD cells is applied if needed.

2.5 Polarisation Shell Models

An atom or ion is polarisable if it develops a dipole moment when placed in an electric field. It is commonly expressed by the equation

$$\underline{\mu} = \alpha \underline{E} \quad , \tag{2.158}$$

where $\underline{\mu}$ is the induced dipole and $\underline{E}$ is the electric field. The constant α is the polarisability.

In the *static* shell model a polarisable atom is represented by a massive core and massless shell, connected by a harmonic spring, hereafter called the core-shell unit. The core and shell carry different electric charges, the sum of which equals the charge on the original atom. There is no

electrostatic interaction (i.e. self interaction) between the core and shell of the same atom. Non-coulombic interactions arise from the shell alone. The effect of an electric field is to separate the core and shell, giving rise to a *polarisation* dipole. The condition of static equilibrium gives the polarisability as:

$$\alpha = (2q_s^2 - q_c^2)/k \quad , \tag{2.159}$$

where q_s and q_c are the shell and core charges and k is the force constant of the harmonic spring.

The calculation of the virial and stress tensor in this model is based on that for a diatomic molecule with charged atoms. The electrostatic and short ranged forces are calculated as described above. The forces of the harmonic springs are calculated as described for intramolecular harmonic bonds. The relationship between the kinetic energy and the temperature is different however, as the core-shell unit is permitted only three translational degrees of freedom, and the degrees of freedom corresponding to rotation and vibration of the unit are discounted as if the kinetic energy of these is regarded as zero (3.11).

2.5.1 Dynamical (Adiabatic Shells)

The dynamical shell model is a method of incorporating polarisability into a molecular dynamics simulation. The method used in DL_POLY_3 is that devised by Fincham *et al* [37] and is known as the adiabatic shell model.

In the adiabatic method, a fraction of the atomic mass is assigned to the shell to permit a dynamical description. The fraction of mass is chosen to ensure that the natural frequency of vibration ν of the harmonic spring (i.e.

$$\nu = \frac{1}{2\pi}\left[\frac{k}{x(1-x)m}\right]^{1/2} \quad , \tag{2.160}$$

with m the atomic mass,) is well above the frequency of vibration of the whole atom in the bulk system. Dynamically the core-shell unit resembles a diatomic molecule with a harmonic bond, however, the high vibrational frequency of the bond prevents effective exchange of kinetic energy between the core-shell unit and the remaining system. Therefore, from an initial condition in which the core-shell units have negligible internal vibrational energy, the units will remain close to this condition throughout the simulation. This is essential if the core-shell unit is to maintain a net polarisation. (In practice there is a slow leakage of kinetic energy into the core-shell units, but this should should not amount to more than a few percent of the total kinetic energy.)

2.5.2 Relaxed (Massless Shells)

The relaxed shell model is presented in [38], where shells have no mass and as such their motion is not governed by the usual Newtonian equation, whereas their cores' motion is. Because of that, shells respond instantaneously to the motion of the cores: for any set of core positions, the positions of the shells are such that the force on every shell is zero. The energy is thus a minimum with respect to the shell positions. This represents the physical fact that the system is always in the ground state with respect to the electronic degrees of freedom.

Relaxation of the shells is carried out at each time step and involves a search in the multidimensional space of shell configurations. The search in DL_POLY_3 is based on the powerful conjugate-gradients technique [39] in an adaptation as shown in [38]. Each time step a few iterations (10÷30) are needed to achieve convergence to zero net force.

In DL_POLY_3 the shell forces are handled by the routine CORE_SHELL_FORCES. In case of the adiabatic shell model the kinetic energy is calculated by CORE_SHELL_KINETIC and temperature

scaling applied by routine CORE_SHELL_QUENCH. In case of the relaxed shell model shell are relaxed to zero force by CORE_SHELL_RELAXED.

Either shell model can be used in conjunction with the methods for long range forces described above.

Note that DL_POLY_3 determines which shell model to use by the supplied shell weights. If all shells have no mass the DL_POLY_3 will choose the relaxed. If all shells have masses then DL_POLY_3 will choose the dynamical. Otherwise if some shells have masses and some do not DL_POLY_3 will terminate execution. At any case DL_POLY_3 prints information about the choice and possible errors in OUTPUT (see Section 5.2.3).

2.6 External Fields

In addition to the molecular force field, DL_POLY_3 allows the use of an *external* force fieldpotential!external field. Examples of fields available include:

1. Electric field: (**elec**)
$$\underline{F_i} = \underline{F_i} + q_i.\underline{H} \tag{2.161}$$
2. Oscillating shear: (**oshm**)
$$\underline{F}_x = A\cos(2n\pi.z/L_z) \tag{2.162}$$
3. Continuous shear: (**shrx**)
$$\underline{v}_x = \frac{1}{2}A\frac{|z|}{z} \qquad : |z| > z_0 \tag{2.163}$$
4. Gravitational field: (**grav**)
$$\underline{F_i} = \underline{F_i} + m_i.\underline{H} \tag{2.164}$$
5. Magnetic field: (**magn**)
$$\underline{F_i} = \underline{F_i} + q_i.(\underline{v_i} \times \underline{H}) \tag{2.165}$$
6. Containing sphere: (**sphr**)
$$\underline{F} = A(R_0 - r)^{-n} \qquad : r > R_{cut} \tag{2.166}$$
7. Repulsive wall: (**zbnd**)
$$\underline{F} = A(z_o - z) \qquad : z > z_o \quad . \tag{2.167}$$

It is recommended that the use of an external field should be accompanied by a thermostat (this does not apply to examples 6 and 7, since these are conservative fields). The "Oscillating shear" field should be used with orthorhombic cell geometry (imcon=1,2) and "Continuous shear" field with slab cell geometry (imcon=6). The user is advised to be careful with units!

In DL_POLY_3 external field forces are handled by the routines EXTERNAL_FIELD_APPLY_VV, EXTERNAL_FIELD_APPLY_VV and EXTERNAL_FIELD_CORRECT.

Chapter 3

Integration Algorithms

Scope of Chapter

This chapter describes the integration algorithms coded into DL_POLY_3 .

3.1 Introduction

As a default the DL_POLY_3 integration algorithms are based on the Velocity Verlet (VV) scheme, which is both simple and time reversible [18]. It generates trajectories in the microcanonical (NVE) ensemble in which the total energy (kinetic plus potential) is conserved. If this property drifts or fluctuates excessively in the course of a simulation it indicates that the timestep is too large or the potential cutoffs too small (relative r.m.s. fluctuations in the total energy of 10^{-5} are typical with this algorithm).

The VV algorithm has two stages (VV1 and VV2). At the first stage it requires values of position ($\underline{r}$), velocity ($\underline{v}$) and force ($\underline{f}$) at time t. The first stage is to advance the velocities to $t + (1/2)\Delta t$ by integration of the force and then to advance the positions to a full step $t + \Delta t$ using the new half-step velocities:

1. VV1:
$$\underline{v}(t + \frac{1}{2}\Delta t) \leftarrow \underline{v}(t) + \frac{\Delta t}{2}\ \frac{\underline{f}(t)}{m} \quad , \tag{3.1}$$
where m is the mass of a site and Δt is the timestep
$$\underline{r}(t + \Delta t) \leftarrow \underline{r}(t) + \Delta t\ \underline{v}(t + \frac{1}{2}\Delta t) \tag{3.2}$$

2. FF:
Between the first and the second stage a recalculation of the force at time $t + \Delta t$ is required since the positions have changed
$$\underline{f}(t) \leftarrow \underline{f}(t + \Delta t) \tag{3.3}$$

3. VV2:
In the second stage the half-step velocities are advanced to to a full step using the new force
$$\underline{v}(t + \Delta t) \leftarrow \underline{v}(t + \frac{1}{2}\Delta t) + \frac{\Delta t}{2}\ \frac{\underline{f}(t + \Delta t)}{m} \tag{3.4}$$

DL_POLY_3 also offers integration algorithms based on the leapfrog Verlet (LFV) scheme [18]. Although LFV scheme is somewhat simpler and numerically faster than the VV scheme, it is not time reversible and does not offer the numerical stability the VV scheme does. Furthermore, properties' estimators are approximate.

The LFV algorithm is one staged. It requires values of position ($\underline{r}$) and force ($\underline{f}$) at time t and velocity ($\underline{v}$) at half a timestep behind: $t - (1/2)\Delta t$. Firstly, the forces are recalculated afresh at time t (from time $t + \Delta t$) since the positions have changed from the last step:

1. FF:
$$\underline{f}(t) \leftarrow \underline{f}(t - \Delta t) \quad , \tag{3.5}$$
where Δt is the timestep

2. LFV:
The velocities are advanced by a timestep to $t + (1/2)\Delta t$ by integration of the new force
$$\underline{v}(t + \frac{1}{2}\Delta t) \leftarrow \underline{v}(t - \frac{1}{2}\Delta t) + \Delta t\ \frac{\underline{f}(t)}{m} \quad , \tag{3.6}$$

where m is the mass of a site, and then the positions are advanced to a full step $t + \Delta t$ using the new half-step velocities

$$\underline{r}(t + \Delta t) \leftarrow \underline{r}(t) + \Delta t \, \underline{v}(t + \frac{1}{2}\Delta t) \tag{3.7}$$

Molecular dynamics simulations normally require properties that depend on position and velocity *at the same time* (such as the sum of potential and kinetic energy). The velocity at time t is obtained from the average of the velocities half a timestep either side of timestep t:

$$\underline{v}(t) \leftarrow \frac{1}{2}\left[\underline{v}(t - \frac{1}{2}\Delta t) + \underline{v}(t + \frac{1}{2}\Delta t)\right] \quad . \tag{3.8}$$

The instantaneous kinetic energy, for example, can then be obtained from the atomic velocities as

$$E_{kin}(t) = \frac{1}{2}\sum_{1}^{\mathcal{N}} m_i v_i^2(t) \quad , \tag{3.9}$$

and assuming the system has no net momentum the instantaneous temperature is

$$\mathcal{T}(t) = \frac{2}{k_B f} E_{kin}(t) \quad , \tag{3.10}$$

where i labels particles, $\mathcal{N}$ the number of particles in the system, k_B the Boltzmann's constant and f the number of degrees of freedom in the system.

$$f = 3\mathcal{N} - 3 - 3\mathcal{N}_{frozen} - 3\mathcal{N}_{shells} - \mathcal{N}_{constraints} - p \quad . \tag{3.11}$$

Here $\mathcal{N}_{frozen}$ indicates the number of frozen atoms in the system, $\mathcal{N}_{shells}$ number of core-shell units and $\mathcal{N}_{constraints}$ number of bond and pmf constraints. Three degrees of freedom are subtracted for the centre of mass zero net momentum (we impose) and p is zero for periodic or three for non-periodic systems.

The routines NVE_1_VV and NVE_1_LFV implement the Verlet algorithm in velocity and leapfrog flavours respectively and calculate the instantaneous temperature. The conserved quantity is the total energy of the system

$$\mathcal{H}_{\rm NVE} = U + E_{kin} \quad , \tag{3.12}$$

where U is the potential energy of the system and E_{kin} the kinetic energy at time t.

The full selection of integration algorithms, indicating both VV and LFV cast integration, within DL_POLY_3 is as follows:

NVE_1_VV	NVE_1_LFV	Constant E algorithm
NVT_L1_VV	NVT_L1_LFV	Constant T algorithm (Langevin [20])
NVT_E1_VV	NVT_E1_LFV	Constant E_{kin} algorithm (Evans [21])
NVT_B1_VV	NVT_B1_LFV	Constant T algorithm (Berendsen [22])
NVT_H1_VV	NVT_H1_LFV	Constant T algorithm (Hoover [23])
NPT_B1_VV	NPT_B1_LFV	Constant T,P algorithm (Berendsen [22])
NPT_H1_VV	NPT_H1_LFV	Constant T,P algorithm (Hoover [23])
NPT_M1_VV	NPT_M1_LFV	Constant T,P algorithm (MTK [24])
NST_B1_VV	NST_B1_LFV	Constant T,$\underline{\underline{\sigma}}$ algorithm (Berendsen [22])
NST_H1_VV	NST_H1_LFV	Constant T,$\underline{\underline{\sigma}}$ algorithm (Hoover [23])
NST_M1_VV	NST_M1_LFV	Constant T,$\underline{\underline{\sigma}}$ algorithm (MTK [24])

3.2 Bond Constraints

The SHAKE algorithm for bond constraints was devised by Ryckaert *et al.* [40] and is widely used in molecular simulation. It is a two stage algorithm based on the leapfrog Verlet integration scheme [18]. In the first stage the LFV algorithm calculates the motion of the atoms in the system assuming a complete absence of the rigid bond forces. The positions of the atoms at the end of this stage do not conserve the distance constraint required by the rigid bond and a correction is necessary. In the second stage the deviation in the length of a given rigid bond is used retrospectively to compute the constraint force needed to conserve the bondlength. It is relatively simple to show that the constraint force has the form

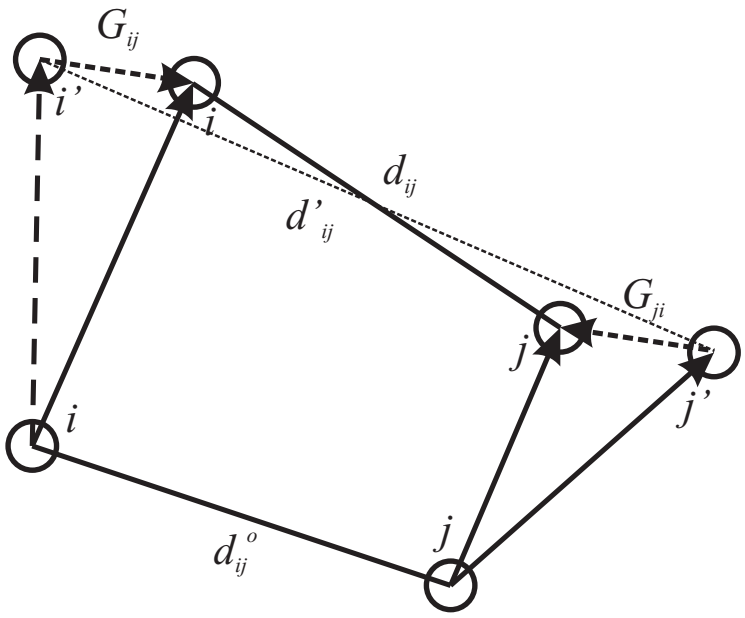
$$\underline{G}_{ij} \approx \frac{1}{2}\frac{\mu_{ij}}{\Delta t^2}\frac{(d_{ij}^2 - d_{ij}'^2)}{\underline{d}_{ij}^o \cdot \underline{d}_{ij}'}\underline{d}_{ij}^o \quad , \tag{3.13}$$

where: μ_{ij} is the reduced mass of the two atoms connected by the bond; $\underline{d}_{ij}^o$ and $\underline{d}_{ij}'$ are the original and intermediate bond vectors; d_{ij} is the constrained bondlength; and Δt is the Verlet integration time step. It should be noted that this formula is an approximation only.

The RATTLE algorithm was devised by Andersen [19] and it fits within the concept of the Velocity Verlet integration scheme. It consists of two parts RATTLE_VV1 and RATTLE_VV2 applied respectively in stages one and two of Velocity Verlet algorithm. RATTLE_VV1 is similar to the SHAKE algorithm as described above and handles the bond length constraint. However, due to the difference in the velocity update between VV (VV1) and LFV schemes, the constraint force generated to conserve the bondlength in RATTLE_VV1 has the form as in (3.13) but missing the factor of a half:

$$\underline{G}_{ij} \approx \frac{\mu_{ij}}{\Delta t^2}\frac{(d_{ij}^2 - d_{ij}'^2)}{\underline{d}_{ij}^o \cdot \underline{d}_{ij}'}\underline{d}_{ij}^o \quad . \tag{3.14}$$

The constraint force in RATTLE_VV2 imposes a new condition of rigidity on constraint bonded atom velocities.



The SHAKE (RATTLE_VV1) algorithm calculates the constraint force $\underline{G}_{ij} = -\underline{G}_{ji}$ that conserves the bondlength d_{ij} between atoms i and j, following the initial movement to positions $i\prime$ and $j\prime$ under the unconstrained forces $\underline{F}_i$ and $\underline{F}_j$ and velocities $\underline{v}_i$ and $\underline{v}_j$.

The RATTLE_VV2, is also a two stage algorithm. In the first stage, the VV algorithm (VV2) calculates the velocities of the atoms in the system assuming a complete absence of the rigid bond forces (since forces have just been recalculated afresh after VV1. The relative velocity of atom i to j (or vice versa) constituting the rigid bond ij is not perpendicular to the bond (i.e. does not have a non-zero component along the bond) as required by the definition of rigidity. In the second stage the deviation from zero of the scalar product $\underline{d}_{ij} \cdot (\underline{v}_j - \underline{v}_i)$ is used retrospectively to compute the constraint force needed to keep the bond rigid over the length of the timestep Δt. It is relatively simple to show that the constraint force has the form:

$$\underline{B}_{ij} \approx \frac{\mu_{ij}}{\Delta t} \frac{\underline{d}_{ij} \cdot (\underline{v}_j - \underline{v}_i)}{d_{ij}^2} \underline{d}_{ij} \quad . \tag{3.15}$$

The velocity corrections can therefore be written as

$$\underline{v}_i^{corr} = \Delta t \frac{\underline{B}_{ij}}{m_i} = \frac{\mu_{ij}}{m_i} \frac{\underline{d}_{ij} \cdot (\underline{v}_j - \underline{v}_i)}{d_{ij}^2} \underline{d}_{ij} \quad . \tag{3.16}$$

For a system of simple diatomic molecules, computation of the constraint force will, in principle, allow the correct atomic positions to be calculated in one pass. However, in the general polyatomic case this correction is merely an interim adjustment, not only because the above formula is approximate, but the successive correction of other bonds in a molecule has the effect of perturbing previously corrected bonds. Either part of the RATTLE algorithm is therefore iterative, with the correction cycle being repeated for all bonds until: each has converged to the correct length, within a given tolerance for RATTLE_VV1 (SHAKE) and the relative bond velocities are perpendicular to their respective bonds within a given tolerance for RATTLE_VV2 (RATTLE). The tolerance may be of the order 10^{-4} Å to 10^{-8} Å depending on the precision desired.

The SHAKE procedure may be summarised as follows:

1. All atoms in the system are moved using the LFV algorithm, assuming an absence of rigid bonds (constraint forces). (This is stage 1 of the SHAKE algorithm.)
2. The deviation in each bondlength is used to calculate the corresponding constraint force (3.13) that (retrospectively) 'corrects' the bond length.
3. After the correction (3.13) has been applied to all bonds, every bondlength is checked. If the largest deviation found exceeds the desired tolerance, the correction calculation is repeated.
4. Steps 2 and 3 are repeated until all bondlengths satisfy the convergence criterion (this iteration constitutes stage 2 of the SHAKE algorithm).

The RATTLE procedures may be summarised as follows:

1. RATTLE stage 1:
   (a) All atoms in the system are moved using the VV algorithm, assuming an absence of rigid bonds (constraint forces). (This is stage 1 of the RATTLE_VV1 algorithm.)
   (b) The deviation in each bondlength is used to calculate the corresponding constraint force (3.14) that (retrospectively) 'corrects' the bond length.
   (c) After the correction (3.14) has been applied to all bonds, every bondlength is checked. If the largest deviation found exceeds the desired tolerance, the correction calculation is repeated.

(d) Steps (b) and (c) are repeated until all bondlengths satisfy the convergence criterion (this iteration constitutes stage 2 of the RATTLE_VV1 algorithm).

2. Forces calculated afresh.

3. RATTLE stage 2:

   (a) All atom velocities are updated to a full step, assuming an absence of rigid bonds. (This is stage 1 of the RATTLE_VV2 algorithm.)

   (b) The deviation of $\underline{d}_{ij} \cdot (\underline{v}_j - \underline{d}_i)$ in each bond is used to calculate the corresponding constraint force that (retrospectively) 'corrects' the bond velocities.

   (c) After the correction (3.15) has been applied to all bonds, every bond velocity is checked against the above condition. If the largest deviation found exceeds the desired tolerance, the correction calculation is repeated.

   (d) Steps (b) and (c) are repeated until all bonds satisfy the convergence criterion (this iteration constitutes stage 2 of the RATTLE_VV2 algorithm).

The parallel version of the RATTLE algorithm, as implemented in DL_POLY_3 , is derived from the RD_SHAKE algorithm [7] although its implementation in the Domain Decomposition framework requires no *global merging* operations and is consequently significantly more efficient. The routine CONSTRAINTS_SHAKE is called to apply corrections to the atomic positions and the routine CONSTRAINTS_RATTLE to apply corrections to the atomic velocities of constrained particles.

It should be noted that the fully converged constraint forces G_{ij} make a contribution to the system virial and the stress tensor.

The contribution to be added to the atomic virial (for each constrained bond) is

$$\mathcal{W} = -\underline{d}_{ij} \cdot \underline{G}_{ij} \quad . \tag{3.17}$$

The contribution to be added to the atomic stress tensor (for each constrained bond) is given by

$$\sigma^{\alpha\beta} = d_{ij}^{\alpha} G_{ij}^{\beta} \quad , \tag{3.18}$$

where α and β indicate the x, y, z components. The atomic stress tensor derived from the pair forces is symmetric.

3.3 Potential of Mean Force (PMF) Constraints and the Evaluation of Free Energy

A generalization of bond constraints can be made to constrain a system to some point along a reaction coordinate. A simple example of such a reaction coordinate would be the distance between two ions in solution. If a number of simulations are conducted with the system constrained to different points along the reaction coordinate then the mean constraint force may be plotted as a function of reaction coordinate and the function integrated to obtain the free energy for the overall process [41]. The PMF constraint force, virial and contributions to the stress tensor are obtained in a manner analagous to that for a bond constraint (see previous section). The only difference is that the constraint is now applied between the centres of two groups which need not be atoms alone. DL_POLY_3 reports the PMF constraint virial, $\mathcal{W}_{PMF}$, for each simulation. Users can convert this to the PMF constraint force from

$$G_{PMF} = \frac{\mathcal{W}_{PMF}}{d_{PMF}} \quad , \tag{3.19}$$

where is d_{PMF} the constraint distance between the two groups used to define the reaction coordinate.

The routines PMF_SHAKE and PMF_RATTLE are called to apply corrections to the atomic positions and respectively the atomic velocities of all particles constituting pmf units.

In presence of both bond constraints and pmf constraints. The constraint procedures, i.e. SHAKE or RATTLE, for both types of constraineds are applied iteratively in order bonds-pmfs until convergence of $\mathcal{W}_{PMF}$ reached. The number of iteration cycles is limited by the same limit as for the constraint procedures.

3.4 Thermostats

The system may be coupled to a heat bath to ensure that the average system temperature is maintained close to the requested temperature, $T_{\rm ext}$. When this is done the equations of motion are modified and the system no longer samples the microcanonical ensemble. Instead trajectories in the canonical (NVT) ensemble, or something close to it are generated. DL_POLY_3 comes with four different thermostats: Langevin [20, 42], Evans (Gaussian constraints) [21], Berendsen [22] and Nosé-Hoover [23]. Of these only the Nosé-Hoover algorithm generates trajectories in the canonical (NVT) ensemble. Evans and Berendsen algorithms will produce properties that typically differ from canonical averages by $\mathcal{O}(1/\mathcal{N})$ [18], as the Evans algorithm generates trajectories in the (NVE$_{kin}$) ensemble. The Langevin method does not generate any proper enesemble at all as it impose Brownian Dynamics (or Stochastic Dynamics) on the system.

3.4.1 Langevin Thermostat

The Langevin thermostat works by coupling every particle to a viscous background and a stochastic heath bath such that

$$\begin{aligned} \frac{d\underline{r}_i(t)}{dt} &= \underline{v}_i(t) \\ \frac{d\underline{v}_i(t)}{dt} &= \frac{\underline{f}_i(t)+\underline{R}_i(t)}{m_i} - \chi\,\underline{v}_i(t) \quad , \end{aligned} \tag{3.20}$$

where χ is a *constant* (user defined in units of ps^{-1}) friction parameter (positive) and $R(t)$ is stochastic force with zero mean that satisfies the fluctuation-dissipation theorem:

$$\left\langle R_i^\alpha(t)\; R_j^\beta(t')\right\rangle = 2\;\chi\;m_i\;k_BT\;\delta_{ij}\;\delta_{\alpha\beta}\;\delta(t-t') \quad , \tag{3.21}$$

where superscripts denote Cartesian indices, subscripts particle indices, k_B is the Boltzmann constant, T the target temperature and m_i the particle's mass. The effect of this algorithm is thermostat the system on a local scale. Particles that are too "cold" are given more energy by the noise term and particles that are too "hot" are slowed down by the friction. Numerical instabilities, which usually arise from inaccurate calculation of a local collision-like process, are thus efficiently kept under control and cannot propagate.

The generation of random forces is implemented in the routine LANGEVIN_FORCES.

The VV implementation the algorithm is tailored in a Langevin Impulse (LI) manner [42]:

1. VV1:

$$\underline{v}(t+\epsilon) \;\leftarrow\; \underline{v}(t) + \frac{\Delta t}{2}\,\frac{\underline{f}(t)}{m}$$

$$\begin{aligned} \underline{v}(t+\frac{1}{2}\Delta t-\epsilon) &\leftarrow \exp(-\chi\Delta t)\,\underline{v}(t+\epsilon)+\frac{\sqrt{2\,\chi\,m\,k_BT}}{m}\,\vec{Z}_1(\chi,\Delta t) \\ \underline{r}(t+\Delta t) &\leftarrow \underline{r}(t)+\frac{1-\exp(-\chi\Delta t)}{\chi}\,\underline{v}(t+\epsilon)+\frac{\sqrt{2\,\chi\,m\,k_BT}}{\chi\,m}\,\vec{Z}_2(\chi,\Delta t)\quad , \end{aligned} \tag{3.22}$$

where $\vec{Z}_1(\chi,\Delta t)$ and $\vec{Z}_2(\chi,\Delta t)$ are joint Gaussian random variables of zero mean, sampling from a *bivariate* Gaussian distribution [42]:

$$\begin{bmatrix} \underline{Z}_1 \\ \underline{Z}_2 \end{bmatrix} = \begin{bmatrix} \sigma_2^{1/2} & 0 \\ (\sigma_1-\sigma_2)\sigma_2^{-1/2} & (\Delta t-\sigma_1^2\sigma_2^{-1})^{1/2} \end{bmatrix} \begin{bmatrix} \underline{R}_1 \\ \underline{R}_2 \end{bmatrix} \tag{3.23}$$

with

$$\sigma_k = \frac{1-\exp(-k\,\chi\,\Delta t)}{k\,\chi}\quad ,\quad k=1,2 \tag{3.24}$$

and $\underline{R}_k$ vectors of independent standard Gaussian random numbers of zero mean and unit variance - easily related to the langevin random forces as defined in equation (3.21)

2. RATTLE_VV1
3. FF:
$$\underline{f}(t+\Delta t) \leftarrow \underline{f}(t) \tag{3.25}$$
4. VV2:
$$\underline{v}(t+\Delta t) \leftarrow \underline{v}(t+\frac{1}{2}\Delta t-\epsilon)+\frac{\Delta t}{2}\,\frac{\underline{f}(t+\Delta t)}{m} \tag{3.26}$$
5. RATTLE_VV2 .

The algorithm is self-consistent and requires no iterations.

The LFV implementation the Langevin algorithm is straightforward:

1. FF:
$$\underline{f}(t) \leftarrow \underline{f}(t-\Delta t) \tag{3.27}$$
2. LFV and Thermostat:
$$\begin{aligned} \underline{v}(t+\frac{1}{2}\Delta t) &\leftarrow \frac{2-\chi\Delta t}{2+\chi\Delta t}\,\underline{v}(t-\frac{1}{2}\Delta t)+\frac{2}{2+\chi\Delta t}\,\Delta t\,\frac{\underline{f}(t)+\underline{R}(t)}{m} \\ \underline{r}(t+\Delta t) &\leftarrow \underline{r}(t)+\Delta t\,\underline{v}(t+\frac{1}{2}\Delta t)\quad , \end{aligned} \tag{3.28}$$

where $\underline{R}(t)$ are Langevin random forces

3. SHAKE
4. Full step velocity:
$$\underline{v}(t) \leftarrow \frac{1}{2}\left[\underline{v}(t-\frac{1}{2}\Delta t)+\underline{v}(t+\frac{1}{2}\Delta t)\right]\quad . \tag{3.29}$$

The VV and LFV flavours of the Langevin algorithm are implemented in the DL_POLY_3 routines NVT_L1_VV and NVT_L1_LFV respectively.

3.4.2 Evans Thermostat (Gaussian Constraints)

Kinetic temperature can be made a constant of the equations of motion by imposing an additional constraint on the system. If one writes the equations of motion as:

$$\begin{aligned} \frac{d\underline{r}(t)}{dt} &= \underline{v}(t) \\ \frac{d\underline{v}(t)}{dt} &= \frac{\underline{f}(t)}{m} - \chi(t)\,\underline{v}(t) \quad , \end{aligned} \tag{3.30}$$

the kinetic temperature constraint χ can be found as follows:

$$\begin{aligned} \frac{d}{dt}\mathcal{T} \;\propto\; & \frac{d}{dt}\left(\frac{1}{2}\sum_i (m_i v_i^2)\right) = \sum_i m_i \underline{v}_i \cdot \frac{d}{dt}\underline{v}_i = 0 \\ & \sum_i m_i \underline{v}_i(t) \cdot \left\{ \frac{\underline{f}_i(t)}{m_i} - \chi(t)\,\underline{v}_i(t) \right\} = 0 \\ \chi(t) &= \frac{\sum_i \underline{v}_i(t) \cdot \underline{f}_i(t)}{\sum_i m_i \underline{v}_i^2(t)} \quad . \end{aligned} \tag{3.31}$$

The VV implementation the algorithm is straight forward. A conventional VV1 and VV2 (thermally unconstrained) steps are carried out. At the end of VV2 $\chi(t+\Delta t)$ is calculated exactly and then the thermal constraint applied on the unthermostated velocities at full step $t+\Delta t$ in the following manner

1. VV1:

$$\begin{aligned} \underline{v}(t+\frac{1}{2}\Delta t) &\leftarrow \underline{v}(t) + \frac{\Delta t}{2}\,\frac{\underline{f}(t)}{m} \\ \underline{r}(t+\Delta t) &\leftarrow \underline{r}(t) + \Delta t\,\underline{v}(t+\frac{1}{2}\Delta t) \end{aligned} \tag{3.32}$$

2. RATTLE_VV1

3. FF:

$$\underline{f}(t+\Delta t) \leftarrow \underline{f}(t) \tag{3.33}$$

4. VV2:

$$\underline{v}(t+\Delta t) \leftarrow \underline{v}(t+\frac{1}{2}\Delta t) + \frac{\Delta t}{2}\,\left[\frac{\underline{f}(t+\Delta t)}{m}\right] \tag{3.34}$$

5. RATTLE_VV2

6. Thermostat:

$$\begin{aligned} \chi(t+\Delta t) &\leftarrow \frac{\sum_i \underline{v}_i(t+\Delta t)\cdot \underline{f}_i(t+\Delta t)}{2\;E_{kin}(t+\Delta t)} \\ \underline{v}(t+\Delta t) &\leftarrow \underline{v}(t+\Delta t)\;\exp\left(-\chi(t+\Delta t)\Delta t\right) \quad . \end{aligned} \tag{3.35}$$

The algorithm is self-consistent and requires no iterations.

The LFV implementation the Evans algorithm is iterative as an initial estimate of $\chi(t)$ at full step is calculated using an unconstrained estimate of the velocity at full step ($\underline{v}(t)$). The iterative part is as follows

1. FF:
$$\underline{f}(t) \leftarrow \underline{f}(t-\Delta t) \quad , \tag{3.36}$$

2. LFV: The iterative part is as follows
$$\begin{aligned} \underline{v}(t+\frac{1}{2}\Delta t) &\leftarrow \frac{2-\chi(t)\Delta t}{2+\chi(t)\Delta t}\,\underline{v}(t-\frac{1}{2}\Delta t) + \frac{2}{2+\chi(t)\Delta t}\,\Delta t\,\frac{\underline{f}(t)}{m} \\ \underline{r}(t+\Delta t) &\leftarrow \underline{r}(t) + \Delta t\,\underline{v}(t+\frac{1}{2}\Delta t) \end{aligned} \tag{3.37}$$

3. SHAKE

4. Full step velocity:
$$\underline{v}(t) \leftarrow \frac{1}{2}\left[\underline{v}(t-\frac{1}{2}\Delta t) + \underline{v}(t+\frac{1}{2}\Delta t)\right] \tag{3.38}$$

5. Thermostat:
$$\chi(t) \leftarrow \frac{\sum_i \underline{v}_i(t)\cdot\underline{f}_i(t)}{2\;E_{kin}(t)} \quad . \tag{3.39}$$

Several iterations are required to obtain self consistency. In DL_POLY_3 the number of iterations is set to 5 (6 if bond constraints are present).

The conserved quantity by these algorithms is the system kinetic energy.

The VV and LFV flavours of the Gaussian constraints algorithm are implemented in the DL_POLY_3 routines NVT_E1_VV and NVT_E1_LFV respectively.

3.4.3 Berendsen Thermostat

In the Berendsen algorithm the instantaneous temperature is pushed towards the desired temperature $T_{\rm ext}$ by scaling the velocities at each step by
$$\chi(t) = \left[1 + \frac{\Delta t}{\tau_T}\left(\frac{\sigma}{E_{kin}(t)} - 1\right)\right]^{1/2} \quad , \tag{3.40}$$
where
$$\sigma = \frac{f}{2}\;k_B\;T_{\rm ext} \tag{3.41}$$
is the target thermostat energy (depending on the external temperature and the system total degrees of freedom, f - equation (3.11)) and τ_T a specified time constant for temperature fluctuations (normally in the range [0.5, 2] ps).

The VV implementation the algorithm is straight forward. A conventional VV1 and VV2 (thermally unconstrained) steps are carried out. At the end of VV2 velocities are scaled by a factor of χ in the following manner

1. VV1:
$$\begin{aligned} \underline{v}(t+\frac{1}{2}\Delta t) &\leftarrow \underline{v}(t) + \frac{\Delta t}{2}\,\frac{\underline{f}(t)}{m} \\ \underline{r}(t+\Delta t) &\leftarrow \underline{r}(t) + \Delta t\,\underline{v}(t+\frac{1}{2}\Delta t) \end{aligned} \tag{3.42}$$

2. RATTLE_VV1

3. FF:
$$\underline{f}(t+\Delta t) \leftarrow \underline{f}(t) \tag{3.43}$$

4. VV2:
$$\underline{v}(t+\Delta t) \leftarrow \underline{v}(t+\frac{1}{2}\Delta t) + \frac{\Delta t}{2}\,\frac{\underline{f}(t+\Delta t)}{m} \tag{3.44}$$

5. RATTLE_VV2

6. Thermostat:
$$\begin{aligned} \chi(t+\Delta t) &\leftarrow \left[1+\frac{\Delta t}{\tau_T}\left(\frac{\sigma}{E_{kin}(t+\Delta t)}-1\right)\right]^{1/2} \\ \underline{v}(t+\Delta t) &\leftarrow \underline{v}(t+\Delta t)\,\chi \quad . \end{aligned} \tag{3.45}$$

The LFV implementation the Berendsen algorithm is iterative as an initial estimate of $\chi(t)$ at full step is calculated using an unconstrained estimate of the velocity at full step, $\underline{v}(t)$.

1. FF:
$$\underline{f}(t) \leftarrow \underline{f}(t-\Delta t) \quad , \tag{3.46}$$

2. LFV: The iterative part is as follows
$$\begin{aligned} \underline{v}(t+\frac{1}{2}\Delta t) &\leftarrow \left[\underline{v}(t-\frac{1}{2}\Delta t) + \Delta t\,\frac{\underline{f}(t)}{m}\right]\,\chi(t) \\ \underline{r}(t+\Delta t) &\leftarrow \underline{r}(t) + \Delta t\,\underline{v}(t+\frac{1}{2}\Delta t) \end{aligned} \tag{3.47}$$

3. SHAKE

4. Full step velocity:
$$\underline{v}(t) \leftarrow \frac{1}{2}\left[\underline{v}(t-\frac{1}{2}\Delta t) + \underline{v}(t+\frac{1}{2}\Delta t)\right] \tag{3.48}$$

5. Thermostat:
$$\chi(t) \leftarrow \left[1+\frac{\Delta t}{\tau_T}\left(\frac{\sigma}{E_{kin}(t)}-1\right)\right]^{1/2} \quad . \tag{3.49}$$

Several iterations are required to obtain self consistency. In DL_POLY_3 the number of iterations is set to 2 (3 if bond constraints are present).

The Berendsen algorithms conserve total momentum but not energy.

The VV and LFV flavours of the Berendsen algorithm are implemented in the DL_POLY_3 routines NVT_B1_VV and NVT_B1_LFV respectively.

3.4.4 Nosé-Hoover Thermostat

In the Nosé-Hoover algorithm [23] Newton's equations of motion are modified to read:
$$\begin{aligned} \frac{d\underline{r}(t)}{dt} &= \underline{v}(t) \\ \frac{d\underline{v}(t)}{dt} &= \frac{\underline{f}(t)}{m} - \chi(t)\,\underline{v}(t) \end{aligned} \tag{3.50}$$

The friction coefficient, χ, is controlled by the first order differential equation

$$\frac{d\chi(t)}{dt} = \frac{2E_{kin}(t) - 2\sigma}{q_{mass}} \tag{3.51}$$

where σ is the target thermostat energy, equation (3.41), and

$$q_{mass} = 2\ \sigma\ \tau_T^2 \tag{3.52}$$

is the thermostat mass, which depends on a specified time constant τ_T (for temperature fluctuations normally in the range [0.5, 2] ps).

The VV implementation the algorithm takes place in symplectic manner as follows:

1. Thermostat: Note $E_{kin}(t)$ changes inside

$$\begin{aligned} \chi(t+\frac{1}{4}\Delta t) &\leftarrow \chi(t) + \frac{\Delta t}{4}\ \frac{2E_{kin}(t) - 2\sigma}{q_{mass}} \\ \underline{v}(t) &\leftarrow \underline{v}(t)\ \exp\left(-\chi(t+\frac{1}{4}\Delta t)\ \frac{\Delta t}{2}\right) \\ \chi(t+\frac{1}{2}\Delta t) &\leftarrow \chi(t+\frac{1}{4}\Delta t) + \frac{\Delta t}{4}\ \frac{2E_{kin}(t) - 2\sigma}{q_{mass}} \end{aligned} \tag{3.53}$$

(3.54)

2. VV1:

$$\begin{aligned} \underline{v}(t+\frac{1}{2}\Delta t) &\leftarrow \underline{v}(t) + \frac{\Delta t}{2}\ \frac{\underline{f}(t)}{m} \\ \underline{r}(t+\Delta t) &\leftarrow \underline{r}(t) + \Delta t\ \underline{v}(t+\frac{1}{2}\Delta t) \end{aligned} \tag{3.55}$$

3. RATTLE_VV1

4. FF:

$$\underline{f}(t+\Delta t) \leftarrow \underline{f}(t) \tag{3.56}$$

5. VV2:

$$\underline{v}(t+\Delta t) \leftarrow \underline{v}(t+\frac{1}{2}\Delta t) + \frac{\Delta t}{2}\ \frac{\underline{f}(t+\Delta t)}{m} \tag{3.57}$$

6. RATTLE_VV2

7. Thermostat: Note $E_{kin}(t+\Delta t)$ changes inside

$$\begin{aligned} \chi(t+\frac{3}{4}\Delta t) &\leftarrow \chi(t+\frac{1}{2}\Delta t) + \frac{\Delta t}{4}\ \frac{2E_{kin}(t+\Delta t) - 2\sigma}{q_{mass}} \\ \underline{v}(t+\Delta t) &\leftarrow \underline{v}(t+\Delta t)\ \exp\left(-\chi(t+\frac{3}{4}\Delta t)\ \frac{\Delta t}{2}\right) \\ \chi(t+\Delta t) &\leftarrow \chi(t+\frac{3}{4}\Delta t) + \frac{\Delta t}{4}\ \frac{2E_{kin}(t+\Delta t) - 2\sigma}{q_{mass}} \quad . \end{aligned} \tag{3.58}$$

The algorithm is self-consistent and requires no iterations.

The LFV implementation the Nosé-Hoover algorithm is iterative as an initial estimate of $\chi(t)$ at full step is calculated using an unconstrained estimate of the velocity at full step, $\underline{v}(t)$.

1. FF:
$$\underline{f}(t) \leftarrow \underline{f}(t-\Delta t) \quad , \tag{3.59}$$

2. LFV: The iterative part is as follows
$$\begin{aligned} \underline{v}(t+\frac{1}{2}\Delta t) &\leftarrow \underline{v}(t-\frac{1}{2}\Delta t) + \Delta t \left[\frac{\underline{f}(t)}{m} - \chi(t)\,\underline{v}(t)\right] \\ \underline{r}(t+\Delta t) &\leftarrow \underline{r}(t) + \Delta t\,\underline{v}(t+\frac{1}{2}\Delta t) \end{aligned} \tag{3.60}$$

3. SHAKE

4. Full step velocity:
$$\underline{v}(t) \leftarrow \frac{1}{2}\left[\underline{v}(t-\frac{1}{2}\Delta t) + \underline{v}(t+\frac{1}{2}\Delta t)\right] \tag{3.61}$$

5. Thermostat:
$$\begin{aligned} \chi(t+\frac{1}{2}\Delta t) &\leftarrow \chi(t-\frac{1}{2}\Delta t) + \Delta t\,\frac{2E_{kin}(t)-2\sigma}{q_{mass}} \\ \chi(t) &\leftarrow \frac{1}{2}\left[\chi(t-\frac{1}{2}\Delta t) + \chi(t+\frac{1}{2}\Delta t)\right] \quad . \end{aligned} \tag{3.62}$$

Several iterations are required to obtain self consistency. In DL_POLY_3 the number of iterations is set to 3 (if no bond constraints are present). If bond constraints are present an extra iteration is required due to the call to the SHAKE routine. Note that the SHAKE corrections need only be applied during the first iteration as subsequent to this the velocities, relative to the bond c.o.m. velocity, will be orthogonal to the bond vectors. The velocity scaling imposed by the thermostat is isotropic so does not destroy this orthogonality.

The conserved quantity is derived from the extended Hamiltonian for the system which, to within a constant, is the Helmholtz free energy:
$$\mathcal{H}_{\rm NVT} = \mathcal{H}_{\rm NVE} + \frac{q_{mass}\,\chi(t)^2}{2} + f\,k_B\,T_{\rm ext}\int_o^t \chi(s)ds \quad , \tag{3.63}$$
where f is the system's degrees of freedom - equation (3.11).

The VV and LFV flavours of the Nosé-Hoover algorithm are implemented in the DL_POLY_3 routines NVT_H1_VV and NVT_H1_LFV respectively.

3.5 Barostats

The size and shape of the simulation cell may be dynamically adjusted by coupling the system to a barostat in order to obtain a desired average pressure ($P_{\rm ext}$) and/or isotropic stress tensor ($\underline{\underline{\sigma}}$). DL_POLY_3 has three such algorithms: the Berendsen barostat [22], the Hoover type barostat [23] and the Martyna-Tuckerman-Klein (MTK) barsotat [24]. Only the last two have a well defined conserved quantities.

3.5.1 Instantaneous pressure and stress

The instantaneous pressure in a system,

$$\mathcal{P}(t) = \frac{[2E_{kin}(t) - \mathcal{W}_{\text{atomic}}(t) - \mathcal{W}_{\text{constrain}}(t-\Delta t) - \mathcal{W}_{\text{PMF}}(t-\Delta t)]}{3V(t)} \quad , \tag{3.64}$$

is a function of the system volume, kinetic energy and virial, $\mathcal{W}$.
Note that when bond constraints or/and PMF constraints are present in the system $\mathcal{P}$ will not converge to the exact value of P_{ext}. This is due to iterative nature of the constrained motion in which the virials $\mathcal{W}_{\text{constrain}}$ and $\mathcal{W}_{\text{PMF}}$ are calculated retrospectively to the forcefield virial $\mathcal{W}_{\text{atomic}}$.

The instantaneous stress tensor in a system,

$$\underline{\underline{\sigma}}(t) = \underline{\underline{\sigma}}_{kin}(t) + \underline{\underline{\sigma}}_{\text{atomic}}(t) + \underline{\underline{\sigma}}_{\text{constrain}}(t-\Delta t) + \underline{\underline{\sigma}}_{\text{PMF}}(t-\Delta t) \quad , \tag{3.65}$$

is a sum of the forcefield, $\underline{\underline{\sigma}}_{\text{atomic}}$, constrain, $\underline{\underline{\sigma}}_{\text{constrains}}$, and PMF, $\underline{\underline{\sigma}}_{\text{PMF}}$, stresses.
Note that when bond constraints or/and PMF constraints are present in the system, the quantity $\frac{\text{Tr}[\underline{\underline{\sigma}}]}{3V}$ will not converge to the exact value of P_{ext}. This is due to iterative nature of the constrained motion in which the constrain and PMF stresses are calculated retrospectively to the forcefield stress.

3.5.2 Berendsen Barostat

With the Berendsen barostat the system is made to obey the equation of motion at the beginning of each step

$$\frac{d\mathcal{P}(t)}{dt} = \frac{P_{\text{ext}} - \mathcal{P}(t)}{\tau_P} \quad , \tag{3.66}$$

where $\mathcal{P}$ is the instantaneous pressure and τ_P is the barostat relaxation time constant.

Cell size variations

In the isotropic implementation, at each step the MD cell volume is scaled by a factor η, and the coordinates and cell vectors by $\eta^{1/3}$, where

$$\eta(t) = 1 - \frac{\beta \Delta t}{\tau_P} \left(P_{\text{ext}} - \mathcal{P}(t)\right) \tag{3.67}$$

and β is the isothermal compressibility of the system. In practice β is a specified constant which DL_POLY_3 takes to be the isothermal compressibility of liquid water. The exact value is not critical to the algorithm as it relies on the ratio τ_P/β. τ_P is a specified time constant for pressure fluctuations, supplied by the user.

In the VV implementation, equation (3.67) imposes an uniform scaling of all cell parameters and particle positions during VV1. No iterations are needed for the VV implementation of this algorithm. However when bond constraints are present in the system 3 iterations are needed in VV1 until satisfactory convergence of the constraint forces (to conserve bond length) is achieved. Note also that the change in box size requires the RATTLE_VV1 algorithm to be called each iteration with the new cell vectors. This algorithm is implemented in NPT_B1_VV.

In the LFV implementation the inclusion of equation (3.67) is by an iterative schema with 4 (3 if bond constraints are present) iterations used to obtain self consistency in the $\underline{v}(t)$. This algorithm is implemented in NPT_B1_LFV.

Cell size and shape variations

The extension of the isotropic algorithm to anisotropic cell variations is straightforward. A tensor $\underline{\underline{\eta}}$ is defined by

$$\underline{\underline{\eta}}(t) = \underline{\underline{\mathbf{1}}} - \frac{\beta\Delta t}{\tau_P}\,\frac{P_{\rm ext}\,V(t)\,\underline{\underline{\mathbf{1}}} - \underline{\underline{\sigma}}(t)}{V(t)} \quad , \tag{3.68}$$

where $\underline{\underline{\mathbf{1}}}$ is the identity matrix. Then new cell vectors are given by

$$\underline{\underline{\mathbf{H}}}(t+\Delta t) \leftarrow \underline{\underline{\eta}}(t)\cdot\underline{\underline{\mathbf{H}}}(t) \quad . \tag{3.69}$$

The VV and LFV flavours of the non-isotropic Berendsen barostat (and thermostat) are implemented in the DL_POLY_3 routines NST_B1_VV and NST_B1_LFV respectively.

3.5.3 The Hoover Barostat

DL_POLY_3 uses the Melchionna modification of the Hoover algorithm [43] in which the equations of motion involve a Nosé-Hoover thermostat and a barostat in the same spirit. Additionally, as shown in [44], a modification allowing for coupling between the thermostat and barostat is also introduced.

Cell size variation

For isotropic fluctuations the equations of motion are:

$$\begin{aligned}
\frac{d}{dt}\underline{r}(t) &= \underline{v}(t) + \eta(t)\,(\underline{r}(t) - \underline{R}_0(t)) \\
\frac{d}{dt}\underline{v}(t) &= \frac{\underline{f}(t)}{m} - [\chi(t)+\eta(t)]\,\underline{v}(t) \\
\frac{d}{dt}\chi(t) &= \frac{2E_{kin}(t) - 2\sigma - k_B\,T_{\rm ext}}{q_{mass}} \\
q_{mass} &= 2\,\sigma\,\tau_T^2 \\
\frac{d}{dt}\eta(t) &= 3V(t)\frac{\mathcal{P}(t) - P_{\rm ext}}{p_{mass}} - \chi(t)\eta(t) \\
p_{mass} &= (f+3)\,k_B\,T_{\rm ext}\,\tau_P^2 \\
\frac{d}{dt}V(t) &= [3\eta(t)]\,V(t) \quad ,
\end{aligned} \tag{3.70}$$

where η is the barostat friction coefficient, $\underline{R}_0(t)$ the system centre of mass at time t, q_{mass} the thermostat mass, τ_T a specified time constant for temperature fluctuations, p_{mass} the barostat mass, τ_P a specified time constant for pressure fluctuations, $\mathcal{P}$ the instantaneous pressure and V the system volume.

The conserved quantity is, to within a constant, the Gibbs free energy of the system:

$$\mathcal{H}_{\rm NPT} = \mathcal{H}_{\rm NVE} + \frac{q_{mass}\,\chi(t)^2}{2} + \frac{p_{mass}\,\eta(t)^2}{2} + P_{\rm ext}V(t) + (f+1)\,k_B\,T_{\rm ext}\int_o^t \chi(s)ds \quad , \tag{3.71}$$

where f is the system's degrees of freedom - equation (3.11).

The VV flavour of the algorithm is implemented in the DL_POLY_3 routine NPT_H1_VV in the following manner:

1. Thermostat: Note $2E_{kin}(t)$ changes inside

$$\begin{aligned}
\chi(t+\frac{1}{8}\Delta t) &\leftarrow \chi(t)+\frac{\Delta t}{8}\,\frac{2E_{kin}(t)+p_{mass}\;\eta(t)^2-2\sigma-k_B\;T_{\rm ext}}{q_{mass}}\\
\underline{v}(t) &\leftarrow \underline{v}(t)\;\exp\left(-\chi(t+\frac{1}{8}\Delta t)\;\frac{\Delta t}{4}\right)\\
\chi(t+\frac{1}{4}\Delta t) &\leftarrow \chi(t+\frac{1}{8}\Delta t)+\frac{\Delta t}{8}\,\frac{2E_{kin}(t)+p_{mass}\;\eta(t)^2-2\sigma-k_B\;T_{\rm ext}}{q_{mass}}
\end{aligned}\tag{3.72}$$

2. Barostat: Note $E_{kin}(t)$ and $\mathcal{P}(t)$ have changed and change inside

$$\begin{aligned}
\eta(t) &\leftarrow \eta(t)\;\exp\left(-\chi(t+\frac{1}{4}\Delta t)\;\frac{\Delta t}{8}\right)\\
\eta(t+\frac{1}{4}\Delta t) &\leftarrow \eta(t)+\frac{\Delta t}{4}\,\frac{3\left[\mathcal{P}(t)-P_{\rm ext}\right]V(t)}{p_{mass}}\\
\eta(t+\frac{1}{4}\Delta t) &\leftarrow \eta(t+\frac{1}{4}\Delta t)\;\exp\left(-\chi(t+\frac{1}{4}\Delta t)\;\frac{\Delta t}{8}\right)\\
\underline{v}(t) &\leftarrow \underline{v}(t)\;\exp\left[-\eta(t+\frac{1}{4}\Delta t)\;\frac{\Delta t}{2}\right]\\
\eta(t+\frac{1}{4}\Delta t) &\leftarrow \eta(t+\frac{1}{4}\Delta t)\;\exp\left(-\chi(t+\frac{1}{4}\Delta t)\;\frac{\Delta t}{8}\right)\\
\eta(t+\frac{1}{2}\Delta t) &\leftarrow \eta(t+\frac{1}{4}\Delta t)+\frac{\Delta t}{4}\,\frac{3\left[\mathcal{P}(t)-P_{\rm ext}\right]V(t)}{p_{mass}}
\end{aligned}\tag{3.73}$$

3. Thermostat: Note $E_{kin}(t)$ has changed and changes inside

$$\begin{aligned}
\chi(t+\frac{3}{8}\Delta t) &\leftarrow \chi(t+\frac{1}{4}\Delta t)+\frac{\Delta t}{8}\,\frac{2E_{kin}(t)+p_{mass}\;\eta(t+\frac{1}{2}\Delta t)^2-2\sigma-k_B\;T_{\rm ext}}{q_{mass}}\\
\underline{v}(t) &\leftarrow \underline{v}(t)\;\exp\left(-\chi(t+\frac{3}{8}\Delta t)\;\frac{\Delta t}{4}\right)\\
\chi(t+\frac{1}{2}\Delta t) &\leftarrow \chi(t+\frac{3}{8}\Delta t)+\frac{\Delta t}{8}\,\frac{2E_{kin}(t)+p_{mass}\;\eta(t+\frac{1}{2}\Delta t)^2-2\sigma-k_B\;T_{\rm ext}}{q_{mass}}
\end{aligned}\tag{3.74}$$

4. VV1:

$$\begin{aligned}
\underline{v}(t+\frac{1}{2}\Delta t) &\leftarrow \underline{v}(t)+\frac{\Delta t}{2}\,\frac{\underline{f}(t)}{m}\\
\underline{\underline{\mathbf{H}}}(t+\Delta t) &\leftarrow \exp\left[\eta(t+\frac{1}{2}\Delta t)\;\Delta t\right]\;\underline{\underline{\mathbf{H}}}(t)\\
V(t+\Delta t) &\leftarrow V(t)\;\exp\left[3\eta(t+\frac{1}{2}\Delta t)\;\Delta t\right]\\
\underline{r}(t+\Delta t) &\leftarrow \left[\frac{V(t+\Delta t)}{V(t)}\right]^{1/3}\;(\underline{r}(t)-\underline{R}_0(t))+\Delta t\;\underline{v}(t+\frac{1}{2}\Delta t)+\underline{R}_0(t)
\end{aligned}\tag{3.75}$$

5. RATTLE_VV1

6. FF:

$$\underline{f}(t+\Delta t)\leftarrow\underline{f}(t)\tag{3.76}$$

7. VV2:

$$\underline{v}(t+\Delta t) \;\leftarrow\; \underline{v}(t) + \frac{\Delta t}{2}\,\frac{\underline{f}(t)}{m} \tag{3.77}$$

8. RATTLE_VV2

9. Thermostat: Note $2E_{kin}(t)$ changes inside

$$\begin{aligned}
\chi(t+\tfrac{5}{8}\Delta t) &\leftarrow \chi(t+\tfrac{1}{2}\Delta t) + \frac{\Delta t}{8}\,\frac{2E_{kin}(t+\Delta t) + p_{mass}\;\eta(t+\frac{1}{2}\Delta t)^2 - 2\sigma - k_B\;T_{\rm ext}}{q_{mass}} \\
\underline{v}(t+\Delta t) &\leftarrow \underline{v}(t+\Delta t)\;\exp\left(-\chi(t+\tfrac{5}{8}\Delta t)\;\frac{\Delta t}{4}\right) \\
\chi(t+\tfrac{3}{4}\Delta t) &\leftarrow \chi(t+\tfrac{5}{8}\Delta t) + \frac{\Delta t}{8}\,\frac{2E_{kin}(t+\Delta t) + p_{mass}\;\eta(t+\frac{1}{2}\Delta t)^2 - 2\sigma - k_B\;T_{\rm ext}}{q_{mass}}
\end{aligned} \tag{3.78}$$

10. Barostat: Note $E_{kin}(t+\Delta t)$ and $\mathcal{P}(t+\Delta t)$ have changed and change inside

$$\begin{aligned}
\eta(t+\tfrac{1}{2}\Delta t) &\leftarrow \eta(t+\tfrac{1}{2}\Delta t)\;\exp\left(-\chi(t+\tfrac{3}{4}\Delta t)\;\frac{\Delta t}{8}\right) \\
\eta(t+\tfrac{3}{4}\Delta t) &\leftarrow \eta(t+\tfrac{1}{2}\Delta t) + \frac{\Delta t}{4}\,\frac{3\left[\mathcal{P}(t+\Delta t) - P_{\rm ext}\right]V(t+\Delta t)}{p_{mass}} \\
\eta(t+\tfrac{3}{4}\Delta t) &\leftarrow \eta(t+\tfrac{3}{4}\Delta t)\;\exp\left(-\chi(t+\tfrac{3}{4}\Delta t)\;\frac{\Delta t}{8}\right) \\
\underline{v}(t+\Delta t) &\leftarrow \underline{v}(t+\Delta t)\;\exp\left[-\eta(t+\tfrac{3}{4}\Delta t)\;\frac{\Delta t}{2}\right] \\
\eta(t+\tfrac{3}{4}\Delta t) &\leftarrow \eta(t+\tfrac{3}{4}\Delta t)\;\exp\left(-\chi(t+\tfrac{3}{4}\Delta t)\;\frac{\Delta t}{8}\right) \\
\eta(t+\Delta t) &\leftarrow \eta(t+\tfrac{3}{4}\Delta t) + \frac{\Delta t}{4}\,\frac{3\left[\mathcal{P}(t+\Delta t) - P_{\rm ext}\right]V(t+\Delta t)}{p_{mass}}
\end{aligned} \tag{3.79}$$

11. Thermostat: Note $E_{kin}(t+\Delta t)$ has changed and changes inside

$$\begin{aligned}
\chi(t+\tfrac{7}{8}\Delta t) &\leftarrow \chi(t+\tfrac{3}{4}\Delta t) + \frac{\Delta t}{8}\,\frac{2E_{kin}(t+\Delta t) + p_{mass}\;\eta(t+\Delta t)^2 - 2\sigma - k_B\;T_{\rm ext}}{q_{mass}} \\
\underline{v}(t+\Delta t) &\leftarrow \underline{v}(t+\Delta t)\;\exp\left(-\chi(t+\tfrac{7}{8}\Delta t)\;\frac{\Delta t}{4}\right) \\
\chi(t+\Delta t) &\leftarrow \chi(t+\tfrac{7}{8}\Delta t) + \frac{\Delta t}{8}\,\frac{2E_{kin}(t+\Delta t) + p_{mass}\;\eta(t+\Delta t)^2 - 2\sigma - k_B\;T_{\rm ext}}{q_{mass}} \\
\underline{v}(t+\Delta t) &\leftarrow \underline{v}(t+\Delta t) - \underline{V}_0(t+\Delta t)\quad,
\end{aligned} \tag{3.80}$$

where$\underline{V}_0(t+\Delta t)$ is the c.o.m. velocity at timestep $t+\Delta t$ and $\underline{\underline{\mathbf{H}}}$ is the cell matrix whose columns are the three cell vectors $\underline{a},\underline{b},\underline{c}$. Like the Nosé-Hoover thermostat, no iterations are required to obtain self consistency. If the system contains constrained bonds then DL_POLY_3 uses 2 iterations as in the first $\mathcal{P}(t+\Delta t)$ and $\underline{f}(t)$ are improved for the second. Note also that the change in box size requires the RATTLE_VV1 algorithm to be called each iteration.

The LFV flavour of the algorithm is implemented in the DL_POLY_3 routine NPT_H1_LFV in iterative manner. It starts with calculation of initial estimates of $\chi(t)$ and $\eta(t)$ at full step by using an unconstrained estimate of the velocity at full step, $\underline{v}(t)$. Also calculated is an unconstrained estimate of the half step position

$$\underline{r}(t+\frac{1}{2}\Delta t) = \frac{1}{2}\left[\underline{r}(t) + \underline{r}(t+\Delta t)\right]\quad. \tag{3.81}$$

1. FF:

$$\underline{f}(t) \leftarrow \underline{f}(t-\Delta t) \quad , \tag{3.82}$$

2. LFV: The iterative part is as follows

$$\begin{aligned}
\underline{v}(t+\frac{1}{2}\Delta t) &\leftarrow \underline{v}(t-\frac{1}{2}\Delta t) + \Delta t \left[\frac{\underline{f}(t)}{m} - (\chi(t)+\eta(t))\,\underline{v}(t)\right] \\
\underline{r}(t+\Delta t) &\leftarrow \underline{r}(t) + \Delta t \left\{\underline{v}(t+\frac{1}{2}\Delta t) - \eta(t+\frac{1}{2}\Delta t)\left[\underline{r}(t+\frac{1}{2}\Delta t) - \underline{R}_0(t)\right]\right\} \\
\underline{\underline{\mathbf{H}}}(t+\Delta t) &\leftarrow \exp\left[\eta(t+\frac{1}{2}\Delta t)\,\Delta t\right]\,\underline{\underline{\mathbf{H}}}(t) \\
V(t+\Delta t) &\leftarrow V(t)\,\exp\left[3\eta(t+\frac{1}{2}\Delta t)\,\Delta t\right]
\end{aligned} \tag{3.83}$$

3. SHAKE

4. Full step velocity and half step position:

$$\begin{aligned}
\underline{v}(t) &\leftarrow \frac{1}{2}\left[\underline{v}(t-\frac{1}{2}\Delta t) + \underline{v}(t+\frac{1}{2}\Delta t)\right] \\
\underline{r}(t+\frac{1}{2}\Delta t) &\leftarrow \frac{\underline{r}(t)+\underline{r}(t+\Delta t)}{2}
\end{aligned} \tag{3.84}$$

5. Thermostat and Barostat:

$$\begin{aligned}
\chi(t+\frac{1}{2}\Delta t) &\leftarrow \chi(t-\frac{1}{2}\Delta t) + \Delta t\,\frac{2E_{kin}(t) + p_{mass}\,\eta(t)^2 - 2\sigma - k_B\,T_{\rm ext}}{q_{mass}} \\
\chi(t) &\leftarrow \frac{1}{2}\left[\chi(t-\frac{1}{2}\Delta t) + \chi(t+\frac{1}{2}\Delta t)\right] \\
\eta(t+\frac{1}{2}\Delta t) &\leftarrow \left\{\eta(t-\frac{1}{2}\Delta t) + \Delta t\,\frac{3\left[\mathcal{P}(t) - P_{\rm ext}\right]V(t)}{p_{mass}}\right\}\,\exp\left(-\chi(t)\Delta t\right) \\
\eta(t) &\leftarrow \frac{1}{2}\left[\eta(t-\frac{1}{2}\Delta t) + \eta(t+\frac{1}{2}\Delta t)\right] \quad .
\end{aligned} \tag{3.85}$$

Several iterations are required to obtain self consistency. In DL_POLY_3 the number of iterations is set to 4 (5 if bond constraints are present). Note also that the change in box size requires the SHAKE algorithm to be called each iteration.

Cell size and shape variation

The isotropic algorithms (VV and LFV) may be extended to allowing the cell shape to vary by defining η as a tensor, $\underline{\underline{\eta}}$. The equations of motion are written in the same fashion as is in the isotropic algorithm with slight modifications (as now the equations for η and V are extended to matrix forms for $\underline{\underline{\eta}}$ and $\underline{\underline{\mathbf{H}}}$)

$$\begin{aligned}
\frac{d}{dt}\chi(t) &= \frac{2E_{kin}(t) - 2\sigma - 3^2\,k_B\,T_{\rm ext}}{q_{mass}} \\
\frac{d}{dt}\underline{\underline{\eta}}(t) &= \frac{\underline{\underline{\sigma}}(t) - P_{\rm ext}\,V(t)\,\underline{\underline{\mathbf{1}}}}{p_{mass}} - \chi(t)\underline{\underline{\eta}}(t) \\
p_{mass} &= \frac{(f+3)}{3}\,k_B\,T_{\rm ext}\,\tau_P^2
\end{aligned} \tag{3.86}$$

$$\begin{aligned}\frac{d}{dt}\underline{\underline{\mathbf{H}}}(t) &= \underline{\underline{\eta}}(t)\cdot\underline{\underline{\mathbf{H}}}(t)\\ \frac{d}{dt}V(t) &= \mathtt{Tr}[\underline{\underline{\eta}}(t)]\ V(t)\ ,\end{aligned}$$

where $\underline{\underline{\sigma}}$ is the stress tensor and $\underline{\underline{\mathbf{1}}}$ is the identity matrix. The VV and LFV algorithmic equations are, therefore, written in the same fashion as above with slight modifications in (i) the equations for the thermostat and barostat frictions, and (ii) the equations for the system volume and cell parameters. The modifications in (i) for the VV couched algorithm read as

$$\begin{aligned}\chi(t+\frac{1}{8}\Delta t) &\leftarrow \chi(t)+\frac{\Delta t}{8}\,\frac{2E_{kin}(t)+p_{mass}\ \mathtt{Tr}[\underline{\underline{\eta}}(t)\cdot\underline{\underline{\eta}}(t)^T]-2\sigma-3^2\ k_B\ T_{\rm ext}}{q_{mass}}\\ \underline{\underline{\eta}}(t+\frac{1}{2}\Delta t) &\leftarrow \underline{\underline{\eta}}(t)+\frac{\Delta t}{2}\,\frac{\underline{\underline{\sigma}}(t)-P_{\rm ext}\ V(t)\ \underline{\underline{\mathbf{1}}}}{p_{mass}}\ ,\end{aligned}\qquad(3.87)$$

whereas for the LFV couched algorithm as

$$\begin{aligned}\chi(t+\frac{1}{2}\Delta t) &\leftarrow \chi(t-\frac{1}{2}\Delta t)+\Delta t\,\frac{2E_{kin}(t)+p_{mass}\ \mathtt{Tr}[\underline{\underline{\eta}}(t)\cdot\underline{\underline{\eta}}(t)^T]-2\sigma-3^2\ k_B\ T_{\rm ext}}{q_{mass}}\\ \underline{\underline{\eta}}(t+\frac{1}{2}\Delta t) &\leftarrow \left[\underline{\underline{\eta}}(t-\frac{1}{2}\Delta t)+\Delta t\,\frac{\underline{\underline{\sigma}}(t)-P_{\rm ext}\ V(t)\ \underline{\underline{\mathbf{1}}}}{p_{mass}}\right]\exp\left(-\chi(t)\Delta t\right)\ .\end{aligned}\qquad(3.88)$$

The modifications in (ii) are the same for both the VV and LFV couched algorithms

$$\begin{aligned}\underline{\underline{\mathbf{H}}}(t+\Delta t) &\leftarrow \exp\left(\underline{\underline{\eta}}(t+\frac{1}{2}\Delta t)\ \Delta t\right)\cdot\underline{\underline{\mathbf{H}}}(t)\\ V(t+\Delta t) &\leftarrow V(t)\ \exp\left(\mathtt{Tr}\left[\underline{\underline{\eta}}(t+\frac{1}{2}\Delta t)\right]\ \Delta t\right)\ .\end{aligned}\qquad(3.89)$$

It is worth noting DL_POLY_3 uses Taylor expansion truncated to the quadratic term to approximate exponentials of tensorial terms.

The conserved quantity is, to within a constant, the Gibbs free energy of the system:

$$\mathcal{H}_{\rm N\underline{\underline{\sigma}}T}=\mathcal{H}_{\rm NVE}+\frac{q_{mass}\ \chi(t)^2}{2}+\frac{p_{mass}\ \mathtt{Tr}[\underline{\underline{\eta}}\cdot\underline{\underline{\eta}}^T]}{2}+P_{\rm ext}V(t)+(f+3^2)\ k_B\ T_{\rm ext}\int_o^t\chi(s)ds\ ,\qquad(3.90)$$

where f is the system's degrees of freedom - equation (3.11).

The VV and LFV flavours of the non-isotropic Nosé-Hoover barostat (and thermostat) are implemented in the DL_POLY_3 routines NST_H1_VV and NST_H1_LFV respectively.

3.5.4 Martyna-Tuckerman-Klein Barostat

DL_POLY_3 includes the Martyna-Tuckerman-Klein (MTK) interpretation of the VV flavoured Hoover algorithms [24] for isotropic and anisotropic cell fluctuations in which the equations of motion are slightly modified to those for the coupled Nosé-Hoover thermostat and barostat so that to exclude the use of R_0 in the position scaling due to the barostat. In this way they are proved to generate ensembles that conserve the phase space volume and thus have well defined conserved quantities even in presence of forces external to the system [44], which is not the case for Nosé-Hoover NPT and N$\underline{\underline{\sigma}}$T ensembles.

The NPT and N$\underline{\underline{\sigma}}$T versions of the MTK ensemble are implemented in the DL_POLY_3 routines NPT_M1_VV and NST_M1_VV (VV flavour), and NPT_M1_LFV and NST_M1_LFV (LFV flavour).

Chapter 4

Construction and Execution

Scope of Chapter

This chapter describes how to compile a working version of DL_POLY_3 and run it.

4.1 Constructing DL_POLY_3 : an Overview

4.1.1 Constructing the Standard Version

DL_POLY_3 was designed as a package of useful subroutines rather than a single program, which means that users are to be able to construct a working simulation program of their own design from the subroutines available, which is capable of performing a specific simulation. However, we recognise that many, perhaps most, users will be content with creating a standard version that covers all of the possible applications and for this reason we have only provided the necessary tools to assemble such a version. The method of creating the standard version is described in detail in this chapter, however a brief step-by-step description follows.

1. DL_POLY_3 is supplied as a UNIX compressed file (tarred and gzipped). This must uncompressed and un-tared to create the DL_POLY_3 directory (Section 1.4).

2. In the *build* subdirectory you will find the required DL_POLY_3 makefiles (see Section 4.2.1 and Appendix C, where the main Makefiles are listed). This must be copied into the subdirectory containing the relevant source code. In most cases this will be the *source* subdirectory.

3. The chosen makefile is executed with an appropriate keyword (Section 4.2.1) which selects for specific parallel platforms. For DL_POLY_3 compilation in parallel mode (Fortran 90 compiler and MPI implementation for the specific machine architecture are required) in many if not most cases users (or the administrators of these platforms) will have to create their own keyword entry in the makefile due to the large variety of (i) software needed for compilation of DL_POLY_3 and (ii) installation paths on different platforms. To facilitate the user with the construction of their own keyword entry, examples are provided in the makefiles. In the case when users use a makefile for DL_POLY_3 compilation in serial mode they will have to provide valid path to the Fortran 90 compiler on the specific platform.

4. The makefile produces the executable version of the code, which as a default will be named DLPOLY.Y and located in the *execute* subdirectory.

5. DL_POLY also has a Java GUI. The files for this are stored in the subdirectory *java*. Compilation of this is simple and requires running the javac compiler and the jar utility. Details for these procedures are provided in the GUI manual [17].

6. To run the executable for the first time you require the files CONTROL, FIELD and CONFIG (and possibly TABLE - if you have tabulated potentials, and REFERENCE - if defect detection is opted in). These must be present in the directory from which the program is executed. (See Section 5.1 for the description of the input files.)

7. Executing the program will produce the files OUTPUT, STATIS, REVCON and REVIVE (and optionally HISTORY, RDFDAT, ZDNDAT, DEFECTS) in the executing directory. (See Section 5.2 for the description of the output files.)

This simple procedure is enough to create a standard version to run most simulations. There may however be some difficulty with array sizes. DL_POLY_3 contains features which allocate arrays after scanning the input files for a simulation. Sometimes these initial estimates are insufficient for a long simulation when, for example, the system volume changes markedly during the simulation or when a system is artificially constructed to have a non-uniform density. Usually, simply restarting the program will cure the problem, but sometimes, especially when the local atom density is a way

higher than the global one, it may be necessary to amend the array sizes in accordance with the error message obtained. A way to trigger lengthening of the density dependent global arrays the user may use the **densvar** option in the CONTROL (Section 5.1.1) file. However, lengthening these array will require a larger amount of memory resources from the execution machine for the simulation, which it may not be able to provide. See Section 6.2.2 for more insight on the DL_POLY_3 source code structure.

4.1.2 Constructing Non-standard Versions

In constructing a non-standard DL_POLY_3 simulation program, the first requirement is for the user to write a program to function as the root segment. The root segment /VV/DL_POLY is placed in the *source* directory and contains the set-up and close-down calls for a molecular dynamics simulation. It is the routine that first opens the OUTPUT file (Section 5.2), which provides the summary of the job. The root program calls the "molecular dynamics cycle" routines /LFV/MD_LFV or /LFV/MD_VV implementing the VV and LFV depending on which integrator has been specified for the simulation. These routines contain major routines required to perform the simulation, control the normal "molecular dynamics cycle" and monitor the *cpu* and *memory* usage. They also bring about a controlled termination of the program; if the *cpu* usage approaches the allotted job time within a pre-set closure time and/or if the *memory* usage approaches the allocated limit for density dependent arrays. Users are recommended to study the forementioned root drives as a model for other implementations of the package they may wish to construct. The dependencies and calling hierarchies of all the DL_POLY_3 subroutines can be found in the Section 6.2.2.

Should additional functionality be added to DL_POLY_3 by the user, the SET_BOUNDS routine (and its support subroutines) may need modifying to allow specification of the dimensions of any new arrays.

Any molecular dynamics simulation performs five different kinds of operation: initialisation; forces calculation; integration of the equations of motion; calculation of system properties; and job termination. It is worth considering these operations in turn and to indicate which DL_POLY_3 routines are available to perform them. We do not give a detailed description, but provide only a guide. Readers are recommended to examine the different routines described in the DL_POLY_3 User Manual for further details (particularly regarding further dependencies i.e. additional routines that may be called).

Initialisation requires firstly that the program determine what platform resources are made available to the the specific simulation job. This is done by the DL_POLY_3 routine MAP_DOMAINS in DOMAINS_MODULE that attempts to allocate and map the resources (nodes in parallel) in compliance with the DD strategy. Next, the routine SET_BOUNDS sets the limits for simulation array sizes and variables to convenient values based on rough scan through the CONFIG, CONTROL, FIELD and optionally TABLE (Section 5.1) files. After allocation of all necessary simulation arrays and variables (compulsory initialisation to zero value), the job control information is required; this is obtained by the routine READ_CONTROL, which reads the CONTROL file. The description of the system to be simulated: the types of atoms and molecules present and the intermolecular forces, are obtained by the READ_FIELD routine, which reads the FIELD file. Lastly, the atomic positions and optionally velocities and forces must be provided. These are obtained by the READ_CONFIG routine, which reads the CONFIG file. The SYSTEM_INIT routine is called next to initialise various simulation arrays and variables intact with the data so far and detects if the job is a restart of previous simulation run. If so it reads the REVOLD (Section 5.1.5) to supply some arrays and variables with the necessary values as saved from the previous job. The domain halo is constructed strait after by the routine SET_HALO_PARTICLES. After gathering all these data, bookkeeping

and exclusion arrays are created for the intramolecular and site related interactions (core_shell, constraint and tether units) by BUILD_BOOK_INTRA and BUILD_EXCL_INTRA routines. Following is the building of the link-cell lists by LINK_CELL_PAIRS, which takes into account the *frozen state* of atoms and the occurrence of atoms in the *excluded atoms* list when building the link-cell lists. Lastly, the thermodynamic properties of the system are checked and set intact by the SET_TEMPERATURE routine (which also generates the initial velocities if required to do so).

The calculation of the pair forces represents the main part of any simulation. A Verlet neighbour list is used by DL_POLY_3 in calculating the atomic forces. The routine that constructs this this from the link-cell lists and calculates all pair forces is called TWO_BODY_FORCES. Variety of subroutines are called within this one to calculate specific contributions by different interactions. For example; VDW_FORCES for the short-range (van der Waals) forces (Section 2.3.1), METAL_LD_COMPUTE and METAL_FORCES for the Sutton-Chen metal interactions (Section 2.3.2), and EWALD_SPME_FORCES, EWALD_REAL_FORCES and EWALD_EXCL_FORCES for the Coulombic forces (Section 2.4).

Higher order intermolecular, site related and intramolecular forces require the routines;
TERSOFF_FORCES, THREE_BODY_FORCES, FOUR_BODY_FORCES,
CORE_SHELL_FORCES or CORE_SHELL_RELAX, TETHERS_FORCES,
BONDS_FORCES, ANGLES_FORCES, DIHEDRALS_FORCES and INVERSIONS_FORCES.
The routines;
EXTERNAL_FIELD_APPLY_VV and EXTERNAL_FIELD_CORRECT, and EXTERNAL_FIELD_APPLY_LFV
are required if the simulated system has an external force field (e.g. electrostatic field) operating.

To help with equilibration simulations, routines such as CAP_FORCES and ZERO_K_OPTIMISE_VV and ZERO_K_OPTIMISE_LFV are sometimes required to reduce the magnitude of badly equilibrated forces.

Integration of the equations of motion is handled by one of the routines listed and described in Chapter 3.

As mentioned elsewhere, DL_POLY_3 does not contain many routines for computing system properties during a simulation. Radial distributions may be calculated however, by using the routines RDF_COLLECT and RDF_COMPUTE. Similarly Z-density distribution may be calculated by using the routines Z_DENSITY_COLLECT and Z_DENSITY_COMPUTE. Ordinary thermodynamic quantities are calculated by the routine STATISTICS_COLLECT, which also writes the STATIS file (Section 5.2.8). Routine TRAJECTORY_WRITE writes the HISTORY (Section 5.2.1) file for later (postmortem) analysis. Routine DEFECTS_WRITE writes the DEFECTS (Section 5.2.2) file for later (postmortem) analysis.

Job termination is handled by the routine STATISTICS_RESULT which writes the final summaries in the OUTPUT file and dumps the restart files REVIVE and REVCON (Sections 5.2.5 and 5.2.4 respectively).

4.2 Compiling and Running DL_POLY_3

4.2.1 Compiling the DL_POLY_3 Source Code

When you have obtained DL_POLY_3 from Daresbury Laboratory and unpacked it, your next task will be to compile it. To aid compilation three general makefiles have been provided in the sub-directory *build*. These are "Makefile_MPI" - for compiling a parallel version of DL_POLY_3 , and "Makefile_SRL1" and "Makefile_SRL2" - for compiling a serial versions (see Appendix C). After choosing what the default compilation is to be, the appropriate makefile is to be copied as

"Makefile" in the sub-directory *source*. The general DL_POLY_3 makefile will build an executable with the full range of functionality - sufficient for the test cases and for most users' requirements. In most cases, the user will have to modify few entries in the specification part of their makefile to match the location of certain software on their system architecture. Note that **only** FORTRAN 90 compiler is required for successful build of DL_POLY_3 in serial mode, and **only** FORTRAN 90 and MPI implementation - for DL_POLY_3 in parallel mode. Should the user add additional functionality to the code, major changes of the makefile may be required!

In UNIX environment the compilation of the program is initiated by typing the command:

make target

where *target* is the specification of the required machine. For many computer systems this is all that is required to compile a working version of DL_POLY_3 . (To determine which targets are already defined in the makefile, examine it or type the command *make* without a nominated target - it will produce a list of known targets.)

The full specification of the *make* command is as follows

make <TARGET= ... > < EX=... > < BINROOT=... >

where some (or all) of the keywords may be omitted. The keywords and their uses are described below. Note that keywords may also be set in the unix environment (e.g. with the "setenv" command in a C-shell).

4.2.1.1 Keywords for the DL_POLY_3 Makefiles

1. **TARGET**
   The TARGET keyword indicates which kind of computer the code is to be compiled for. This **must** be specified - there is no default value. Valid targets can be listed by the makefile if the command *make* is typed, without arguments. The list frequently changes as more targets are added and redundant ones removed. Users are encouraged to extend the makefile for themselves, using existing targets as examples.

2. **EX**
   The EX keyword specifies the executable name. The default name for the executable is "DLPOLY.Y".

3. **BINROOT**
   The BINROOT keyword specifies the directory in which the executable is to be stored. The default setting is "../execute".

4.2.1.2 Modifying the DL_POLY_3 Makefiles

1. **Changing the FORTRAN 90 compiler and MPI implementation**
   To specify the FORTRAN 90 compiler in a target platform, the user must type the full path to the executable in **FC="..."** and all appropriate options (as defined in the relevant FORTRAN 90 manual) and the path to the MPI implementation in **FCFLAGS="..."**. The same must be done for the the linker: the path to the executable in **LD="..."** and the appropriate options and the path to the MPI implementation in **LDFLAGS="..."**.

2. **Adding new functionality**
   To include a new subroutine in the code simply add *subroutine*.o to the list of object names in the makefile ('OBJ_ALL"). Note that there is a hierarchial order of adding file names in the "OBJ_MOD" list whereas such order does not exist in the "OBJ_ALL" list. Therefore, should dependence exist between routines listed in the "OBJ_ALL" list, it **must** be explicitly declared in the makefile.

4.2.1.3 Note on the DL_POLY_3 Interpolation Scheme

In DL_POLY_3 the some short-range-like contributions (Van der Waals, Sutton Chen, real space Ewald summation) to energy and force are evaluated by interpolation of tables constructed at the beginning of execution. The DL_POLY_3 interpolation scheme is based on a 3-point linear interpolation in r.

The number of grid points (`mxgrid`) required for interpolation in r to give good energy conservation in a simulation is:

$$\texttt{mxgrid} = \texttt{Max}(1000, \texttt{Int}(r_{\texttt{cut}}/0.01 + 0.5) + 4) \quad ,$$

where $r_{\texttt{cut}}$ is the main cutoff beyond which the contributions from the short-range-like interactions are negligible.

4.2.2 Running DL_POLY_3

To run the DL_POLY_3 executable (DLPOLY.Y) you will initially require three, possibly four, input data files, which you must create in the *execute* sub-directory, (or whichever sub-directory you keep the executable program). The first of these is the CONTROL file (Section 5.1.1), which indicates to DL_POLY_3 what kind of simulation you want to run, how much data you want to gather and for how long you want the job to run. The second file you need is the CONFIG file (Section 5.1.2). This contains the atom positions and, depending on how the file was created (e.g. whether this is a configuration created from 'scratch' or the end point of another run), the velocities and forces also. The third file required is the FIELD file (Section 5.1.3), which specifies the nature of the intermolecular interactions, the molecular topology and the atomic properties, such as charge and mass. Sometimes you will require a fourth file: TABLE (Section 5.1.6), which contains the potential and force arrays for functional forms not available within DL_POLY_3 (usually because they are too complex e.g. spline potentials) and/or a fifth file: REFERENCE (Section 5.1.4), which is similar to the CONFIG and contains the "perfect" crystalline structure of the system.

Examples of input files are found in the *data* sub-directory, which can be copied into the *execute* subdirectory using the *select* macro found in the *execute* sub-directory.

A successful run of DL_POLY_3 will generate several data files, which appear in the *execute* sub-directory. The most obvious one is the file OUTPUT (Section 5.2.3), which provides an effective summary of the job run: the input information; starting configuration; instantaneous and rolling-averaged thermodynamic data; final configurations; radial distribution functions (RDFs); and job timing data. The OUTPUT file is human readable. Also present will be the restart files REVIVE (Section 5.2.5) and REVCON (Section 5.2.4). REVIVE contains the accumulated data for a number of thermodynamic quantities and RDFs, and is intended to be used as the input file for a following run. It is *not* human readable. The REVCON file contains the *restart configuration* i.e. the final positions, velocities and forces of the atoms when the run ended and is human readable. The STATIS file (Section 5.2.8) contains a catalogue of instantaneous values of thermodynamic and other variables, in a form suitable for temporal or statistical analysis. Finally, the HISTORY file

(Section 5.2.1) provides a time ordered sequence of configurations to facilitate further analysis of the atomic motions. By default this file is formatted (human readable) but with little effort from the user it can be generated unformatted. You may move these output files back into the *data* sub-directory using the *store* macro found in the *execute* sub-directory.

Lastly, DL_POLY_3 may also create the files RDFDAT, ZDNDAT and DEFECTS containing the RDF, Z-density and defects data respectively. They are all human readable files.

4.2.3 Restarting DL_POLY_3

The best approach to running DL_POLY_3 is to define from the outset precisely the simulation you wish to perform and create the input files specific to this requirement. The program will then perform the requested simulation, but may terminate prematurely through error, inadequate time allocation or computer failure. Errors in input data are your responsibility, but DL_POLY_3 will usually give diagnostic messages to help you sort out the trouble. Running out of job time is common and provided you have correctly specified the job time variables (using the **close time** and **job time** directives - see Section 5.1.1) in the CONTROL file, DL_POLY_3 will stop in a controlled manner, allowing you to restart the job as if it had not been interrupted.

To restart a simulation after normal termination you will again require the CONTROL file, the FIELD (and TABLE) file, and a CONFIG file, which is the exact copy of the REVCON file created by the previous job. You will also require a new file: REVOLD (Section 5.1.5), which is an exact copy of the previous REVIVE file. If you attempt to restart DL_POLY_3 without this additional file available, the job will most probably fail. Note that DL_POLY_3 will append new data to the existing STATIS and HISTORY files if the run is restarted, other output files will be **overwritten**.

In the event of machine failure, you should be able to restart the job in the same way from the surviving REVCON and REVIVE files, which are dumped at regular intervals to meet just such an emergency. In this case check carefully that the input files are intact and use the HISTORY and STATIS files with caution - there may be duplicated or missing records. The reprieve processing capabilities of DL_POLY_3 are not foolproof - the job may crash while these files are being written for example, but they can help a great deal. You are advised to keep backup copies of these files, noting the times they were written, to help you avoid going right back to the start of a simulation.

You can also extend a simulation beyond its initial allocation of timesteps, provided you still have the REVCON and REVIVE files. These should be copied to the CONFIG and REVOLD files respectively and the directive **timesteps** adjusted in the CONTROL file to the new total number of steps required for the simulation. For example if you wish to extend a 10000 step simulation by a further 5000 steps use the directive **timesteps 15000** in the CONTROL file and include the **restart** directive.

You can use the **restart scale** directive if you want to reset the temperature at the restart or **restart noscale** directive if you want to keep the current kinetics intact, but note that the latter makes no use of REVOLD and all internal accumulators are reset to zero.

If none of the restart options is specified velocities are generated anew with Gaussian distribution of the target kinetic energy based on the provided temperature in the CONTROL file.

4.2.4 Notes on the DL_POLY_3 Simulation Efficiency and Performance

Although the DL_POLY_3 underlining parallelisation strategy (DD and link-cells, see Section 6.1.1) is extremely efficient, it cannot always provide linear parallelisation speed gain with increasing pro-

cessor count for a fixed size system. Nevertheless, it will always provide speedup of the simulation (i.e. there still is a sufficient speed gain in simulations when the number of nodes used in parallel is increased). The simplest explanation why this is is that increasing the processor count for a fixed size system decreases not only the work- and memory-load per processor but also the ratio size of domain to size of halo (both in counts of link cells). When this ratio falls down to values close to one and below, the time DL_POLY_3 spends on inevitable communication (MPI messages across neighbouring domains to refresh the halo data) increases with respect to and eventually becomes prevalent to the time DL_POLY_3 spends on numeric calculations (integration and forces). In such regimes, the **overall** DL_POLY_3 efficiency falls down since processors stay idle and no power is harnessed while communications are not finished.

It is important that the user recognises when DL_POLY_3 becomes vulnerable to decreased efficiency and what possible measures could be taken to avoid this. DL_POLY_3 calculates and reports the link-cell algorithms ($M_x \cdot M_y \cdot M_z$) employed in the simulations immediately after execution. M_x (analogously for M_y and M_z) is the integer number of the ratio of the width of the system domains in x-direction to the maximal short-range cutoff specified for the system:

$$M_x \;=\; \mathtt{Nint}\left[\frac{\mathtt{Max}(x_{ij})/\#(\mathtt{nodes})_{x-\mathtt{direction}}}{\mathtt{Max}(\mathtt{cutoff})}\right] \quad . \tag{4.1}$$

Every domain (node) of the MD cell is loaded with $(M_x + 2) \cdot (M_y + 2) \cdot (M_z + 2)$ link-cells of which $M_x \cdot M_y \cdot M_z$ belong to that domain and the rest are a halo image of link-cells from neighbouring domains. In this respect, the more linked cells per domain, the less halo data to keep, the more efficient the load distribution per node and the less the communications between nodes and therefore, the better the parallelisation.

DL_POLY_3 issues a built-in warning when a link-cell algorithms has a dimension less than four (i.e. less than four link-cells per domain in given direction). A useful rule of thumb is that parallelisation speedup inefficiency is expected when the ratio:

$$\mathtt{R} = \frac{M_x \cdot M_y \cdot M_z}{(M_x + 2) \cdot (M_y + 2) \cdot (M_z + 2) - M_x \cdot M_y \cdot M_z}$$

is close to or drops below one. In such cases there are three strategies for improving the situation that can be used singly or in combination. As obvious from equation (4.1) these are; **(i)** decrease the number of nodes used in parallel, **(ii)** decrease the cutoff and **(iii)** increase system size. It is crucial to note that increased parallelisation efficiency remains even when the link-cell algorithm is used inefficiently. However, DL_POLY_3 will issue an error message and cease execution if detects it cannot fit a link-cell per domain as this is the minimum the DL_POLY_3 link-cell algorithm can work with - $(1 \cdot 1 \cdot 1)$ corresponding to ratio $\mathtt{R} = 1/26$.

Performance may also affected by the fluctuations in the inter-node communication, due to unavoidable communication traffic when a simulation job does not have exclusive use of all machine resources. Such effects may worsen the performance much especially when the average calculation time is of the same magnitude as or less than the average communication time (i.e. nodes spend more time communicating rather than calculating).

4.3 A Guide to Preparing Input Files

The CONFIG file and the FIELD file can be quite large and unwieldy particularly if a polymer or biological molecule is involved in the simulation. This section outlines the paths to follow when trying to construct files for such systems. The description of the DL_POLY_3 force field in

Chapter 2 is essential reading. The various utility routines mentioned in this section are described in greater detail in the DL_POLY_2 Reference Manual. Many of these have been incorporated into the DL_POLY_3 Graphical User Interface [17] and may be conveniently used from there.

4.3.1 Inorganic Materials

The utility GENLAT can be used to construct the CONFIG file for relatively simple lattice structures. Input is interactive. The FIELD file for such systems are normally small and can be constructed by hand. Otherwise, the input of force field data for crystalline systems is particularly simple, if no angular forces are required (notable exceptions to this are zeolites and silicate glasses - see below). Such systems require only the specification of the atomic types and the necessary pair forces. The reader is referred to the description of the DL_POLY_3 FIELD file for further details (Section 5.1.3).

DL_POLY_3 can simulate zeolites and silicate (or other) glasses. Both these materials require the use of angular forces to describe the local structure correctly. In both cases the angular terms are included as *three body terms*, the forms of which are described in Chapter 2. These terms are entered into the FIELD file with the pair potentials.

An alternative way of handling zeolites is to treat the zeolite framework as a kind of macromolecule (see below). Specifying all this is tedious and is best done computationally: what is required is to determine the nearest image neighbours of all atoms and assign appropriate bond and valence angle potentials. What must be avoided at all costs is specifying the angle potentials *without* specifying bond potentials. In this case DL_POLY_3 will automatically cancel the non-bonded forces between atoms linked via valence angles and the system will collapse. The advantage of this method is that the calculation is likely to be faster than using three-body forces. This method is not recommended for amorphous systems.

4.3.2 Macromolecules

Simulations of proteins are best tackled using the package DLPROTEIN [45] which is an adaptation of DL_POLY specific to protein modelling. However, you may simulate proteins and other macromolecules with DL_POLY_3 if you wish. This is described below.

If you select a *protein* structure from a SEQNET file (e.g. from the Brookhaven database), use the utility PROSEQ to generate the file CONFIG. This will then function as input for DL_POLY_3 . Some caution is required here however, as the protein structure may not be fully determined and atoms may be missing from the CONFIG file.

If you have the "edit.out" file produced by AMBER for your molecule use this as the CONNECT_DAT input file for the utility AMBFORCE. AMBFORCE will produce the DL_POLY_3 FIELD and CONFIG files for your molecule.

If you do not have the "edit.out" file things are a little more tricky, particularly in coming up with appropriate partial charges for atomic sites. However, there are a series of utilities that will at least produce the CONNECT_DAT file for use with AMBFORCE. We now outline these utilities and the order in which they should be used.

If you have a structure from the Cambridge Structural database (CSDB) then use the utility FRACCON to take fractional coordinate data and produce a CONNECT_DAT and "ambforce.dat" file for use with AMBFORCE. Note that you will need to modify FRACCON to get the AMBER names correct for sites in your molecule. The version of FRACCON supplied with DL_POLY_3 is specific

to the valinomycin molecule.

If you require an all atom force field and the database file does not contain hydrogen positions then use the utility FRACFILL in place of FRACCON. FRACCON produces an output file HFILL which should then be used as input for the utility HFILL. The HFILL utility fills out the structure with the missing hydrogens. (Note that you may need to know what the atomic charges are in some systems, for example the AMBER charges from the literature.)

Note: with minor modifications the utilities FRACFILL and FRACCON can be used on structures from databases other than the Cambridge structural database.

4.3.3 Adding Solvent to a Structure

The utility WATERADD adds water from an equilibrated configuration of 256 SPC water molecules at 300 K to fill out the MD cell. The utility SOLVADD fills out the MD box with single-site solvent molecules from a fcc lattice. The FIELD files will then need to be edited to account for the solvent molecules added to the file.

Hint: to save yourself some work in entering the non-bonded interactions variables involving solvent sites to the FIELD file put two bogus atoms of each solvent type at the end of the CONNECT_DAT file (for AMBER force-fields) the utility AMBFORCE will then evaluate all the non-bonded variables required by DL_POLY_3 . Remember to delete the bogus entries from the CONFIG file before running DL_POLY_3 .

4.3.4 Analysing Results

DL_POLY_3 is not designed to calculate every conceivable property you might wish from a simulation. Apart from some obvious thermodynamic quantities and radial distribution functions, it does not calculate anything beyond the atomic trajectories. You must therefore be prepared to post-process the HISTORY file if you want other information. There are some utilities in the DL_POLY_3 package to help with this, but the list is far from exhaustive. In time, we hope to have many more. Our users are invited to submit code to the DL_POLY_3 *public* library to help with this.

The utilities available are described in the DL_POLY_2 Reference Manual. Users should also be aware that many of these utilities are incorporated into the DL_POLY Graphical User Interface [17].

4.3.5 Choosing Ewald Sum Variables

4.3.5.1 Ewald sum and SPME

This section outlines how to optimise the accuracy of the Smoothed Particle Mesh Ewald sum parameters for a given simulation..

As a guide to beginners DL_POLY_3 will calculate reasonable parameters if the **ewald precision** directive is used in the CONTROL file (see Section 5.1.1). A relative error (see below) of 10^{-6} is normally sufficient so the directive

ewald precision 1d-6

will make DL_POLY_3 evaluate its best guess at the Ewald parameters α, `kmax1`, `kmax2` and `kmax3`. (The user should note that this represents an *estimate*, and there are sometimes circumstances where the estimate can be improved upon. This is especially the case when the system contains a strong directional anisotropy, such as a surface.) These four parameters may also be set explicitly by the **ewald sum** directive in the CONTROL file. For example the directive

ewald sum 0.35 6 6 8

would set $\alpha = 0.35$ Å^{-1}, `kmax1` $= 6$, `kmax2` $= 6$ and `kmax3` $= 8$. The quickest check on the accuracy of the Ewald sum is to compare the coulombic energy (U) and virial ($\mathcal{W}$) in a short simulation. Adherence to the relationship $U = -\mathcal{W}$, shows the extent to which the Ewald sum is correctly converged. These variables can be found under the columns headed `eng_cou` and `vir_cou` in the OUTPUT file (see Section5.2.3).

The remainder of this section explains the meanings of these parameters and how they can be chosen. The Ewald sum can only be used in a three dimensional periodic system. There are five variables that control the accuracy: α, the Ewald convergence parameter; $r_{\rm cut}$ the real space force cutoff; and the `kmax1,2,3` integers that specify the dimensions of the SPME charge (and FFT) array. The `kmax1,2,3` integers effectively define the range of the reciprocal space sum (one integer for each of the three axis directions). These variables are not independent, and it is usual to regard one of them as pre-determined and adjust the others accordingly. In this treatment we assume that $r_{\rm cut}$ (defined by the **cutoff** directive in the CONTROL file) is fixed for the given system.

The Ewald sum splits the (electrostatic) sum for the infinite, periodic, system into a damped real space sum and a reciprocal space sum. The rate of convergence of both sums is governed by α. Evaluation of the real space sum is truncated at $r = r_{\rm cut}$ so it is important that α be chosen so that contributions to the real space sum are negligible for terms with $r > r_{\rm cut}$. The relative error (ϵ) in the real space sum truncated at $r_{\rm cut}$ is given approximately by

$$\epsilon \approx \mathrm{erfc}(\alpha\ r_{\rm cut})/r_{\rm cut} \approx \exp[-(\alpha\ r_{\rm cut})^2]/r_{\rm cut} \tag{4.2}$$

The recommended value for α is $3.2/r_{\rm cut}$ or greater (too large a value will make the reciprocal space sum very slowly convergent). This gives a relative error in the energy of no greater than $\epsilon = 4 \times 10^{-5}$ in the real space sum. When using the directive **ewald precision** DL_POLY_3 makes use of a more sophisticated approximation:

$$\mathrm{erfc}(x) \approx 0.56\ \exp(-x^2)/x \tag{4.3}$$

to solve recursively for α, using equation 4.2 to give the first guess.

The relative error in the reciprocal space term is approximately

$$\epsilon \approx \exp(-k_{max}^2/4\alpha^2)/k_{max}^2 \tag{4.4}$$

where

$$k_{max} = \frac{2\pi}{L}\ \frac{\texttt{kmax}}{2} \tag{4.5}$$

is the largest k-vector considered in reciprocal space, L is the width of the cell in the specified direction and `kmax` is an integer.

For a relative error of 4×10^{-5} this means using $k_{max} \approx 6.2\ \alpha$. `kmax` is then

$$\texttt{kmax} > 6.4\ L/r_{\rm cut}. \tag{4.6}$$

In a cubic system, $r_{\rm cut} = L/2$ implies `kmax` = 14. In practice the above equation slightly over estimates the value of `kmax` required, so optimal values need to be found experimentally. In the above example `kmax` = 10 or 12 would be adequate.

4.4 DL_POLY_3 Warning and Error Processing

4.4.1 The DL_POLY_3 Internal Warning Facility

DL_POLY_3 contains a number of various in-built checks scattered throughout the package which detect a range of possible inconsistencies or errors. In all cases, such a check fails the subroutine WARNING is called, resulting in an appropriate message that identifies the inconsistency. In some cases an inconsistency is resolved by DL_POLY_3 supplying a default value or DL_POLY_3 assuming a priority of one directive over the another (in clash of mutually exclusive directives). However, in other cases this cannot be done and controlled termination of the program execution is called by the subroutine ERROR. In any case appropriate diagnostic message is displayed notifying the user of the nature of the problem.

4.4.2 The DL_POLY_3 Internal Error Facility

DL_POLY_3 contains a number of in-built error checks scattered throughout the package which detect a wide range of possible errors. In all cases, when an error is detected the subroutine ERROR is called, resulting in an appropriate message and termination of the program execution (either immediately, or after some additional processing). In some case, if the cause for error is considered to be mendable it is corrected and the subroutine WARNING results in an appropriate message.

Users intending to insert new error checks should ensure that all error checks are performed *concurrently* on *all* nodes, and that in circumstances where a different result may obtain on different nodes, a call to the global status routine GCHECK is made to set the appropriate global error flag on all nodes. Only after this is done, a call to subroutine ERROR may be made. An example of such a procedure might be:

```
Logical ::  safe
safe = (test_condition)
Call gcheck(safe)
If (.not.safe) Call error(message_number)
```

In this example it is assumed that the logical operation *test_condition* will result in the answer *.true.* if it is safe for the program to proceed, and *.false.* otherwise. The call to ERROR requires the user to state the `message_number` is an integer which used to identify the appropriate message to be printed.

A full list of the DL_POLY_3 error messages and the appropriate user action can be found in Appendix D of this document.

Chapter 5

Data Files

Scope of Chapter

This chapter describes all the input and output files for DL_POLY_3 , examples of which are to be found in the *data* sub-directory.

5.1 The INPUT Files

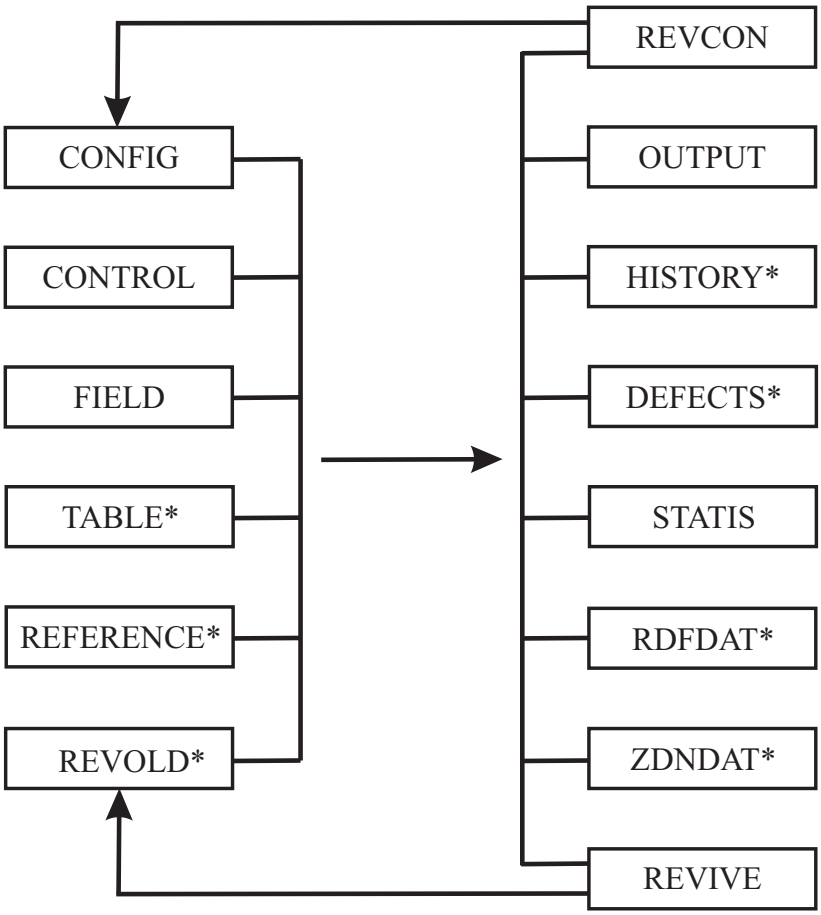


Figure 5.1: DL_POLY_3 input (left) and output (right) files. Note: files marked with an asterisk are non-mandatory.

DL_POLY_3 requires six input files named CONTROL, CONFIG, FIELD, TABLE REFERENCE and REVOLD. The first three files are mandatory, whereas TABLE is used only to input certain kinds of pair potentials, and is not always required. REFERENCE is required only if defect detection is switched on in CONTROL. REVOLD is required only if the job represents a continuation of a previous job. In the following sections we describe the form and content of these files.

5.1.1 The CONTROL File

The CONTROL file is read by the subroutine READ_CONTROL and defines the control variables for running a DL_POLY_3 job. (It is also read by the subroutine SCAN_CONTROL in the SET_BOUNDS routine.) It makes extensive use of **directives** and **keywords**. Directives are character strings that appear as the first entry on a data record (or line) and which invoke a particular operation or provide numerical parameters. Also associated with each directive may be one or more keywords, which may qualify a particular directive by, for example, adding extra options. Directives can appear in any order in the CONTROL file, except for the **finish** directive which marks the end of the file. Some of the directives are mandatory (for example the **timestep** directive that defines

the timestep), others are optional.

This way of constructing the file is very convenient, but it has inherent dangers. It is, for example, quite easy to specify contradictory directives, or invoke algorithms that do not work together. By large DL_POLY_3 tries to sort out these difficulties and print helpful error messages, but it does not claim to be foolproof. Another common mistake is to specify more than once a directive that has no contradictory, disabling, altering or antagonistic directives - then the one specified last will be used as a control directive (for example **densvar, equil, steps, press, mxshak,tolnce, ...**). Fortunately, in most cases the CONTROL file will be small and easy to check visually. It is important to think carefully about a simulation beforehand and ensure that DL_POLY_3 is being asked to do something that is physically reasonable. It should also be remembered that the present capabilities the package may not allow the simulation required and it may be necessary for you yourself to add new features.

An example CONTROL file appears below. The directives and keywords appearing are described in the following section.

```
Title Record: Example CONTROL file for DL_POLY_3

# state parameters
# (target) temperature
temperature            300.0 (Kelvin)
# (target) pressure
pressure                10.0 (katms)

# integration options
# ensemble and options
ensemble npt hoover      0.1 1.0 (ps)
# integration timestep
timestep                 0.0005 (ps)

# simulation options
# full length of simulation
steps                 2000
# length of equilibration
equilibration         1000 (steps)
# temperature scaling interval (during equilibration)
scale                    5 (steps)
# force capping option (during equilibration)
cap                   2000 (kT/A)

# cutoffs
# long-ranged interactions cutoff
cut                      7.6 (A)
# short-ranged interactions cutoff
rvdw                     4.6 (A)

# ewald summation parameters
ewald sum                0.48  32 32 32

# statistics controls
```

```
# print controller for OUTPUT
print                  100 (steps)
# statistics averaging interval
stack                  100 (deep)
# statistics collection interval for STATIS
stats                 2000 (steps)

# rdf sampling option
rdf                     10 (steps)
# rdf print controller for OUTPUT and RDFDAT
print rdf

# trajectory sampling controls for HISTORY
# start step, step interval, level of information
trajectory  1  50   0

# job time and permitted wind-up time
job time             36000 (s)
close time              20 (s)

finish
```

5.1.1.1 The CONTROL File Format

The file is free-formatted and not case-sensitive. Every line is treated as a command sentence (record). Commented records (beginning with a #) and blank lines are not processed and may be added to aid legibility (see example above). Records must be limited in length to 100 characters. Records are read in words (**directives** and additional **keywords** and **numbers**), as a word must not exceed 40 characters in length. Words are recognised as such by separation by one or more space characters. Additional annotation is not recommended but may be added onto a directive line after the last control word in it.

- The first record in the CONTROL file is a header (up to 100 characters long) to aid identification of the file.
- The last record is a **finish** directive, which marks the end of the input data.

Between the header and the **finish** directive, a wide choice of control directives may be inserted. These are described below.

5.1.1.2 The CONTROL File Directives

The directives available are as follows:

directive:	**meaning:**
cap (forces) f	cap forces during equilibration period, f is maximum cap in units of $k_BT/Å$ (default $f = 1000$ $k_BT/Å$)
close time f	set job closure time to f seconds

collect	include equilibration data in overall statistics
coulomb	calculate electrostatic forces using direct Coulomb sum
cutoff f	set required long-ranged interactions cutoff to f Å
defects i j f	write DEFECTS file with controls: i = start timestep for dumping defects configurations (default $i = 0$) j = timestep interval between configurations (default $j = 1$) f = site-interstitial cutoff (default $f = 0.75$ Å, see Table 5.1)
densvar f	allow for local variation in system particle density of $\approx f$ % (useful for extremely non-equilibrium simulations, default $f = 0$)
distance	calculate electrostatic forces using Coulomb sum with distance dependent dielectric
dump n	set restart data dump interval to n steps
ensemble nve	select NVE ensemble (default ensemble)
ensemble nvt ber f	select NVT ensemble with Berendsen thermostat with relaxation constant f in ps
ensemble nvt evans	select NVE$_{kin}$ ensemble with Evans thermostat (Gaussian constraints)
ensemble nvt hoover f	select NVT ensemble with Hoover-Nose thermostat with with relaxation constant f in ps
ensemble nvt lang f	select NVT ensemble with Langevin thermostat with relaxation speed (friction) constant f in ps$_{-1}$
ensemble npt ber f_1 f_2	select Berendsen NPT ensemble with f_1, f_2 as the thermostat and barostat relaxation times in ps
ensemble npt hoover f_1 f_2	select Hoover NPT ensemble with f_1, f_2 as the thermostat and barostat relaxation times in ps
ensemble npt mtk f_1 f_2	select Martyna-Tuckerman-Klein NPT ensemble with f_1, f_2 as the thermostat and barostat relaxation times in ps
ensemble nst ber f_1 f_2	select Berendsen N$\underline{\underline{\sigma}}$T ensemble with f_1, f_2 as the thermostat and barostat relaxation times in ps
ensemble nst hoover f_1 f_2	select Hoover N$\underline{\underline{\sigma}}$T ensemble with f_1, f_2 as the thermostat and barostat relaxation times in ps
ensemble nst mtk f_1 f_2	select Martyna-Tuckerman-Klein NσT ensemble with f_1, f_2 as the thermostat and barostat relaxation times in ps
epsilon (constant) f	set relative dielectric constant to f (default $f = 1.0$)
equilibration **(steps)** n	equilibrate simulation for first n timesteps
ewald precision f	calculate electrostatic forces using Ewald sum with automatic parameter optimisation ($0 < f < 0.5$)
ewald (sum) α k_1 k_2 k_3	calculate electrostatic forces using Ewald sum with α = Ewald convergence parameter in Å^{-1} $k1$ = is twice the maximum k-vector index in x-direction $k2$ = is twice the maximum k-vector index in y-direction $k3$ = is twice the maximum k-vector index in z-direction
exclude	switches on extended coulombic exclusion affecting intra-molecular interactions such as: chemical bonds and bond angles; as well as bond constraints between ions that have shells and cores
finish	close the CONTROL file (last data record)
integrator **(string)**	set the type of Verlet integrator, where **(string)** can only be **(leapfrog)** or **(velocity)**, the later being the default

job time f	set job time to f seconds
maxdis f	set maximum distance allowed in variable timestep (control) to f Å(default $f = 0.10$ Å)
mindis f	set minimum distance allowed in variable timestep (control) to f Å(default $f = 0.03$ Å)
mxshak f	set shake/rattle iterations limit to f (default $f = 250$)
no elec	ignore electrostatics in simulation
no vdw	ignore short range (non-bonded) interactions in simulation
pressure f	set required system pressure to f katms (target pressure for constant pressure ensembles)
print (every) n	print system data every n timesteps
print rdf	print radial distribution functions
print zden	print Z-density profile
pseudo f	enclose the MD cell in a pseudo thermal bath, f is the thickness of the thermostat layers attached on the inside at the MD cell boundaries in units of Å (default $f = 2$ Å)
rdf (sampling) (every) f	calculate and collect radial distribution functions every f timesteps (default $f = 1$)
reaction	calculate electrostatic forces using reaction field electrostatics
restart	restart job from end point of previous run (i.e. continue current simulation, REVOLD required)
restart noscale	restart job from previous run without scaling system temperature (i.e. begin a new simulation from older run without temperature reset, REVOLD is not used)
restart scale	restart job from previous run with scaling system temperature (i.e. begin a new simulation from older run with temperature reset, REVOLD is not used)
rlxtol f	set tolerance for relaxed shell model to f (default $f = 1$ in D Å ps^{-2})
rvdw (cutoff) f	set required short-ranged interactions cutoff to f Å
scale n	rescale atomic velocities every n steps during equilibration (with respect to the last equilibration step)
shake (tolerance) f	set shake/rattle tolerance to f (default $f = 10^{-6}$)
shift	calculate electrostatic forces using force shifted Coulomb sum
stack (size) n	set rolling average stack to n timesteps
stats (every) n	accumulate statistics data every n timesteps
steps n	run simulation for n timesteps
temperature f	set required simulation temperature to f Kelvin (target temperature for constant temperature ensembles)
trajectory i j k	write HISTORY file with controls: i = start timestep for dumping configurations (default $i = 0$) j = timestep interval between configurations (default $j = 1$) k = data level (default $k = 0$, see Table 5.1)
timestep f	set timestep to f ps
variable timestep f	variable timestep, start with timestep of f ps
zden (sampling) (every) f	calculate and collect the Z-density profile every f timesteps (default $f = 1$)
zero	perform zero temperature MD run (reset required system temperature 10 Kelvin)

Note that in some cases additional keywords, shown in brackets "(...)", may also be supplied in the directives, or directives may be used in a long form. However, it is strongly recommended that the user uses only the **bold** part of these directives.

Table 5.1: Internal Trajectory/Defects File Key

keytrj	meaning
0	coordinates only in file
1	coordinates and velocities in file
2	coordinates, velocities and forces in file

5.1.1.3 Further Comments on the CONTROL File

1. A number of the directives (or their **mutually exclusive** alternatives) are **mandatory**:
   (a) **timestep** or **variable timestep**: specifying the simulation timestep
   (b) **temp** or **zero**: specifying the system temperature (not mutually exclusive).
   (c) **ewald sum** or **ewald precision** or **coul** or **shift** or **distan** or **reaction** or **no elec**: specifying the required coulombic forces option
   (d) **cut**: specifying the short range forces cutoff.
2. Some directives are optional. If not specified DL_POLY_3 will take default values if necessary. The defaults are specified above in the list of directives.
3. The **variable timestep** (or also **timestep variable**) option requires the user to specify an initial guess for a reasonable timestep for the system (in picoseconds). The simulation is unlikely to retain this as the operational timestep however, as the latter may change in response to the dynamics of the system. The option is used in conjunction with the default values of **maxdis** (0.03 Å) and **mindis** (0.10 Å), which can also be optionally altered if used as directives (note the rule that **maxdis** $>$ 2.5 **mindis** applies). These distances serve as control values in the variable timestep algorithm, which calculates the greatest distance a particle has travelled in any timestep during the simulation. If the maximum distance is exceeded, the timestep variable is halved and the step repeated. If the greatest move is less than the minimum allowed, the timestep variable is doubled and the step repeated. In this way the integration timestep self-adjusts in response to the dynamics of the system.
4. The **job time** and **close time** directives are required to ensure a controlled close down procedure when a job runs out of time. The time specified by the **job time** directive indicates the total time allowed for the job. (This must obviously be set equal to the time specified to the operating system when the job is submitted.) The **close time** directive represents the time DL_POLY_3 will require to write and close all the data files at the end of processing. This means the *effective* processing time limit is equal to the job time minus the close time. Thus when DL_POLY_3 reaches the effective job time limit it begins the close down procedure with enough time in hand to ensure the files are correctly written. In this way you may be sure the restart files etc. are complete when the job terminates. Note that setting the close time too small will mean the job will crash before the files have been finished. If it is set too large DL_POLY_3 will begin closing down too early. How large the close time needs to be to ensure safe close down is system dependent and a matter of experience. It generally increases with increasing simulation system size.

5. The starting options for a simulation are governed by the keyword **restart**. If this is **not** specified in the control file, the simulation will start as new. If specifed, it will either continue a previous simulation (**restart**) or start a new simulation without initial temperature scaling of the previous configuration (**restart noscale**). Internally these options are handled by the integer variable `keyres`, which is explained in Table 5.2.

Table 5.2: Internal Restart Key

`keyres`	meaning
0	start new simulation from CONFIG file and assign velocities from Gaussian distribution
1	continue current simulation
2	start new simulation from CONFIG file and rescale velocities to desired temperature
3	start new simulation from CONFIG file and **do not** rescale velocities

6. The various **ensemble** options (i.e. **nve**, **nvt** lang, **nvt** evans, **nvt ber**, **nvt hoover**, **npt ber**, **npt hoover**, **npt mtk**, **nst ber**, **nst hoover**, **nst mtk**) are mutually exclusive, though none is mandatory (the default is the NVE ensemble). These options are handled internally by the integer variable `keyens`. The meaning of this variable is explained in Table 5.3.

Table 5.3: Internal Ensemble Key

`keyens`	meaning
0	Microcanonical ensemble (NVE)
1	Langevin NVT ensemble
2	Evans NVT ensemble
3	Berendsen NVT ensemble
4	Nosé-Hoover NVT ensemble
5	Berendsen NPT ensemble
6	Nosé-Hoover NPT ensemble
7	Martyna-Tuckerman-Klein NPT ensemble
8	Berendsen N$\underline{\underline{\sigma}}$T ensemble
9	Nosé-Hoover N$\underline{\underline{\sigma}}$T ensemble
10	Martyna-Tuckerman-Klein NσT ensemble

7. The **zero** directive, enables a "zero temperature" optimisation. The target temperature of the simulation is reset to 10 Kelvin and a crude energy minimiser:

$$\underline{v}_i(t) \leftarrow \begin{cases} 0 & : \quad \underline{v}_i(t) \cdot \underline{f}_i(t) < 0 \\ \underline{f}_i(t) \dfrac{\underline{v}_i(t) \cdot \underline{f}_i(t)}{\underline{f}_i(t) \cdot \underline{f}_i(t)} & : \quad \underline{v}_i(t) \cdot \underline{f}_i(t) \geq 0 \end{cases} \tag{5.1}$$

is used to help the system relax before each integration of the equations of motion (measures taken to conserve the MD cell momentum). This must not be thought of as a true energy minimization method. Note that this optimisation is applied irrespectively of whether the simulation runs in equilibration or statistical mode.

The algorithm is developed in the DL_POLY_3 routines ZERO_K_OPTIMISE_VV and ZERO_K_OPTIMISE_LFV respectively.

8. The **pseudo** option is intended to be used in highly non-equilibrium simulations when users are primarily interested in the structural changes in the core of the simulated system as the the MD cell boundaries of the system are coupled to a Langevin bath (stochastic thermostat) with temperature the target temperature of the system. The stochasticity of the thermostat emulates an infinite environment around the MD cell, providing a means for "natural" heat exchange between the MD system and the heath bath thus aiding possible heat build up in the system. In this way the instantaneous temperature of the system is driven naturally towards the target temperature. The user is required to specify the width of the pseudo thermostat, f (in Å), which must be larger than 2 Å and less than or equal to a quarter of minimum width of the MD cell. The thermostat is an f Å thick buffer layer attached on the inside at the MD cell boundaries. Every particle within the buffer is coupled to a viscous background and a stochastic heat bath, such that

$$\begin{aligned} \frac{d\underline{r}_i(t)}{dt} &= \underline{v}_i(t) \\ \frac{d\underline{v}_i(t)}{dt} &= \frac{\underline{f}_i(t) + \underline{R}_i(t)}{m_i} - \chi(t)\ \underline{v}_i(t) \quad , \end{aligned} \tag{5.2}$$

where $\chi(t)$ is the friction parameter from the dynamics in the the MD cell and $R(t)$ is stochastic force with zero mean that satisfies the fluctuation-dissipation theorem:

$$\left\langle R_i^\alpha(t)\ R_j^\beta(t') \right\rangle = 2\ \chi(t)\ m_i\ k_B T\ \delta_{ij}\ \delta_{\alpha\beta}\ \delta(t-t') \quad , \tag{5.3}$$

where superscripts denote Cartesian indices, subscripts particle indices, k_B is the Boltzmann constant, T the target temperature and m_i the particle's mass. The algorithm is implemented in routine PSEUDO and has two stages:

- Generate random forces on all particles within the thermostat. Here, care must be exercised to prevent introduction of non-zero net force when the random forces are added to the system force field.
- Rescale the kinetic energy of the thermostat bath so that particles within have Gaussian distributed kinetic energy with respect to the target temperature and determine the (Gaussian constraint) friction within the thermostat:

$$\chi(t) = Max\left(0, sum_i frac[\underline{f}_i(t) + \underline{R}_i(t)]/dotvecv_i(t) m_i\ vecv_i^2(t)\right) \quad . \tag{5.4}$$

  Care must be exercised to prevent introduction of non-zero net momentum. (Users are reminded to use for target temperature the temperature at which the original system was equilibrated in order to avoid simulation instabilities.)

The effect of this algorithm is to relax the buffer region of the system on a local scale and to effectively dissipate the incoming excess kinetic energy from the rest of the system, thus emulating an infinite-like environment surrounding the MD cell. The thermostat width matters as the more violent the events on the inside of the MD cell, the bigger width may be needed in order to ensure safe dissipation of the excess kinetic energy.

Note that embedding a thermostat in the MD cell walls is bound to produce **wrong ensemble averages**, and instantaneous pressure and stress build-ups at the thermostat boundary.

Therefore, ensembles lose their meaning as such and so does the conserved quantity for true ensembles.

The algorithm is developed in the DL_POLY_3 routines PSEUDO_VV and PSEUDO_LFV respectively.

9. The **defe**cts option will trigger reading of REFERENCE (see Section 5.1.4), which defines a reference MD cell with particles' positions defining the crystalline lattice sites. If REFERENCE is not found the simulation will either
(i) holt if the simulation has been restarted, i.e. is a continuation of an old one - the **restart** option is used in CONTROL and the REVOLD (see Section 5.1.5) file has been provided.
or
(ii) recover using CONFIG (see Section 5.1.2) if it is a new simulation run, i.e **restart** option is not used in CONTROL or REVOLD has not been provided.

   The actual defect detection is based on comparison of the simulated MD cell to the reference MD cell based on a user defined site-interstitial cutoff, 0.3 Å $\leq$ R_{def} $\leq$ 1.2 Å (default value 0.75 Å, if the supplied value exceeds the limits the simulation execution will holt). If a particle, p, is located in the vicinity of a site, s, defined by a sphere with its centre at this site and a radius, R_{def}, then the particle is a *first hand* claimee of s, and the site is not vacant. Otherwise, the site is presumed vacant and the particle is presumed a general interstitial. If a site, s, is claimed and another particle, $p\prime$, is located within the sphere around it, then $p\prime$ becomes an interstitial associated with s. After all particles and all sites are considered, it is clear which sites are vacancies. Finally, for every claimed site, distances between the site and its *first hand* claimee and interstitials are compared and the particle with the shortest one becomes the *real* claimee. If a *first hand* claimee of s is not the *real* claimee it becomes an interstitial associated with s. At this stage it is clear which particles are interstitials. The sum of interstitials and vacancies gives the total number of defects in the simulated MD cell.

   Frozen particles and particles detected to be shells of polarisable ions are not considered in the defect detection.

   Note that the algorithm cannot be applied safely if R_{def} is larger than half the shortest interatomic distance within the reference MD cell since a particle may; (i) claim more than one site, (ii) be an interstitial associated with more than one site, or both (i) and (ii). On the other hand, low values of R_{def} are likely to lead to slight overestimation of defects.

   If the simulation and reference MD cell have the same number of atoms then the total number of interstitials is always equal to the total number of defects.

10. The tolerance for relaxed shell model **rlxtol**, is a last resort option to aid shell relaxation of systems with very energetic and/or rough potential surface. Users are advised to use it with caution, should there really need be, as the use of high values may result in physically incorrect dynamics.

11. The force selection directives **ewald sum**, **ewald precision**, **reaction**, **coul**, **shift**, **dist**, **no elec** are handled internally by the integer variable `keyfce`. See Table 5.4 for an explanation of this variable. Note that these options are mutually exclusive.

12. The choice of reaction field electrostatics (directive **reaction**) requires also the specification of the relative dielectric constant external to the cavity. This is specified in the **eps** directive.

13. DL_POLY_3 uses two different potential cutoffs. These are as follows:

Table 5.4: Electrostatics Key

keyfce	meaning
Electrostatics are evaluated as follows:	
0	Ignore electrostatic interactions
2	SPME Ewald summation
4	Coulomb sum with distance dependent dielectric
6	Standard truncated Coulomb sum
8	Force shifted Coulomb sum
10	Reaction field electrostatics

(a) $r_{\texttt{cut}}$ - the universal cutoff set by **cutoff**. It applies to the real space part of the electrostatics calculations and to the van der Waals potentials if no other cutoff is applied.

(b) $r_{\texttt{vdw}}$ - the user-specified cutoff for the van der Waals potentials set by **rvdw**. If not specified its value defaults to $r_{\texttt{cut}}$.

14. Constraint algorithms in DL_POLY_3 , SHAKE/RATTLE (see Section 3.2), use default iteration precision of 10^{-8} and limit of iteration cycles of 250. Users may experience that during optimisation of a new built system containing constraints simulation may fail prematurely since a constraint algorithm failed to converge. In such cases directives **mxshak** (to increase) and **shake** (to decrease) may be used to decrease the strain in the system and stablise the simulation numerics until equilibration is achieved.

15. DL_POLY_3 's DD strategy assumes that the global density of a system does not vary much during a simulation, which is not the case in extremely non-equilibrium simulations or some specific systems. A way to tackle such circumstances and avoid simulations crash is to use the **densvar** f option. In the SET_BOUNDS subroutine DL_POLY_3 makes assumptions at the beginning of the simulation and corrects the lengths of link-cell (`mxlist`) and domain (`mxatms`) lists arrays when the option is activated with $f > 0$. Greater values of f will correspond to allocation bigger global arrays and larger memory consumption by DL_POLY_3 during the simulation. Note that this option may demand more memory than available on the computer architecture. In such cases DL_POLY_3 will terminate with an array allocation failure message.

16. As a default, DL_POLY_3 does not store statistical data during the equilibration period. If the directive **collect** is used, equilibration data will be incorporated into the overall statistics.

Users are advised to study the example CONTROL files appearing in the *data* sub-directory to see how different files are constructed.

5.1.2 The CONFIG File

The CONFIG file contains the dimensions of the unit cell, the key for periodic boundary conditions and the atomic labels, coordinates, velocities and forces. This file is read by the subroutine READ_CONFIG. (It is also read by the subroutine SCAN_CONFIG in the SET_BOUNDS routine.) The first few records of a typical CONFIG file are shown below:

```
IceI structure 6x6x6 unit cells with proton disorder
```

```
         2         3
  26.988000000000000   0.000000000000000   0.000000000000000
 -13.494000000000000  23.372293600000000   0.000000000000000
   0.000000000000000   0.000000000000000  44.028000000000000
      OW         1
    -2.505228382         -1.484234330         -7.274585343
     0.5446573999        -1.872177437        -0.7702718106
      3515.939287          13070.74357          4432.030587
      HW         2
    -1.622622646         -1.972916834         -7.340573742
      1.507099154        -1.577400769          4.328786484
      7455.527553         -4806.880540        -1255.814536
      HW         3
    -3.258494716         -2.125627191         -7.491549620
      2.413871957        -4.336956694          2.951142896
    -7896.278327         -8318.045939        -2379.766752
      OW         4
    0.9720599243E-01     -2.503798635         -3.732081894
      1.787340483        -1.021777575         0.5473436377
      9226.455153          9445.662860          5365.202509

etc.
```

5.1.2.1 The CONFIG File Format

The file is free-formatted and not case sensitive. Every line is treated as a command sentence (record). However, line records are limited to 72 characters in length. Records are read in words, as a word must not exceed 40 characters in length. Words are recognised as such by separation by one or more space characters. The first record in the CONFIG file is a header (up to 72 characters long) to aid identification of the file. Blank and commented lines are not allowed.

5.1.2.2 Definitions of Variables in the CONFIG File

record 1		
`header`	a72	title line
record 2		
`levcfg`	integer	CONFIG file key. See Table 5.5 for permitted values
`imcon`	integer	Periodic boundary key. See Table 5.6 for permitted values
record 3	omitted if `imcon` = 0	
`cell(1)`	real	x component of a cell vector
`cell(2)`	real	y component of a cell vector
`cell(3)`	real	z component of a cell vector
record 4	omitted if `imcon` = 0	
`cell(4)`	real	x component of b cell vector
`cell(5)`	real	y component of b cell vector
`cell(6)`	real	z component of b cell vector
record 5	omitted if `imcon` = 0	
`cell(7)`	real	x component of c cell vector
`cell(8)`	real	y component of c cell vector

`cell(9)`	real	z component of c cell vector

Note that **record 2** may contain more information apart from the mandatory as listed in it, any other items are not read by DL_POLY_3 but may be added to aid alternative uses of the file.

Subsequent records consists of blocks of between 2 and 4 records depending on the value of the `levcfg` variable. Each block refers to one atom. The atoms **do not** need to be listed sequentially in order of increasing index. Within each block the data are as follows:

record i		
`atmnam`	a8	atom name
`index`	integer	atom index
record ii		
`xxx`	real	x coordinate
`yyy`	real	y coordinate
`zzz`	real	z coordinate
record iii	included only if `levcfg` > 0	
`vxx`	real	x component of velocity
`vyy`	real	y component of velocity
`vzz`	real	x component of velocity
record iv	included only if `levcfg` > 1	
`fxx`	real	x component of force
`fyy`	real	y component of force
`fzz`	real	z component of force

Note that on **record i** only the atom name is strictly mandatory, any other items are not read by DL_POLY_3 but may be added to aid alternative uses of the file, for example the DL_POLY_3 Graphical User Interface[17].

Table 5.5: CONFIG File Key (record 2)

`levcfg`	meaning
0	coordinates included in file
1	coordinates and velocities included in file
2	coordinates, velocities and forces included in file

Table 5.6: Periodic Boundary Key (record2)

`imcon`	meaning
0	no periodic boundaries
1	cubic boundary conditions
2	orthorhombic boundary conditions
3	parallelepiped boundary conditions
6	x-y parallelogram boundary conditions with no periodicity in the z direction

5.1.2.3 Further Comments on the CONFIG File

The CONFIG file has the same format as the output file REVCON (Section 5.2.4). When restarting from a previous run of DL_POLY_3 (i.e. using the **restart**, **restart noscale** or **restart scale** directives in the CONTROL file - above), the CONFIG file must be replaced by the REVCON file, which is renamed as the CONFIG file. The *copy* macro in the *execute* sub-directory of DL_POLY_3 does this for you.

5.1.3 The FIELD File

The FIELD file contains the force field information defining the nature of the molecular forces. It is read by the subroutine READ_FIELD. (It is also read by the subroutine SCAN_FIELD in the SET_BOUNDS routine.) Excerpts from a force field file are shown below. The example is the antibiotic Valinomycin in a cluster of 146 water molecules.

```
Valinomycin Molecule with 146 SPC Waters
UNITS kcal

MOLECULES    2
Valinomycin
NUMMOLS 1
ATOMS 168
    O         16.0000     -0.4160          1
    OS        16.0000     -0.4550          1
    "           "           "        "
    "           "           "        "
    HC         1.0080      0.0580          1
    C         12.0100      0.4770          1
BONDS 78
harm   31   19 674.000     1.44900
harm   33   31 620.000     1.52600
    "    "    "    "          "
    "    "    "    "          "
harm  168   19 980.000     1.33500
harm  168  162 634.000     1.52200
CONSTRAINTS 90
   20   19    1.000017
   22   21    1.000032
    "    "     "
    "    "     "
  166  164    1.000087
  167  164    0.999968
ANGLES 312
harm   43    2   44  200.00      116.40
harm   69    5   70  200.00      116.40
  "    "    "    "     "           "
  "    "    "    "     "           "
harm   18  168  162  160.00      120.40
harm   19  168  162  140.00      116.60
```

```
DIHEDRALS 371
harm    1   43    2   44  2.3000      180.00
harm   31   43    2   44  2.3000      180.00
  "    "    "    "    "     "           "
  "    "    "    "    "     "           "
cos   149   17  161   16  10.500      180.00
cos   162   19  168   18  10.500      180.00
FINISH
SPC Water
NUMMOLS 146
ATOMS  3
    OW       16.0000     -0.8200
    HW        1.0080      0.4100
    HW        1.0080      0.4100
CONSTRAINTS  3
    1    2    1.0000
    1    3    1.0000
    2    3    1.63299
FINISH
VDW   45
C       C         lj   0.12000      3.2963
C       CT        lj   0.08485      3.2518
"       "         "     "            "
"       "         "     "            "
"       "         "     "            "
OW      OS        lj   0.15100      3.0451
OS      OS        lj   0.15000      2.9400
CLOSE
```

5.1.3.1 The FIELD File Format

The file is free-formatted and not case-sensitive. Every line is treated as a command sentence (record). Commented records (beginning with a #) and blank lines are not processed and may be added to aid legibility (see example above). Records must be limited in length to 100 characters. Records are read in words, as a word must not exceed 40 characters in length. Words are recognised as such by separation by one or more space characters. The contents of the file are variable and are defined by the use of **directives**. Additional information is associated with the directives.

5.1.3.2 Definitions of Variables in the FIELD File

The file divides into three sections: general information, molecular descriptions, and non-bonded interaction descriptions, appearing in that order in the file.

5.1.3.2.1 General information The first viable record in the FIELD file is the title. The second is the **units** directive. Both of these are mandatory.

record 1

header	a100	field file header

record 2

units	a40	Unit of energy used for input and output

The energy units on the **units** directive are described by additional keywords:

a. eV, for electron-volts per mol

b. kcal, for k-calories per mol

c. kJ, for k-Joules per mol

d. internal, for DL_POLY internal units (10 Joules per mol).

If no units keyword is entered, DL_POLY internal units are assumed for both input and output. The **units** directive only affects the input and output interfaces, all internal calculations are handled using DL_POLY units. System output energies are read in **units per MD cell**.

5.1.3.2.2 Molecular details It is important for the user to understand that there is an organisational correspondence between the FIELD file and the CONFIG file described above. It is required that the order of specification of molecular types and their atomic constituents in the FIELD file follows the order of indices in which they appear in the CONFIG file. Failure to adhere to this common sequence will be detected by DL_POLY_3 and result in premature termination of the job. It is therefore essential to work from the CONFIG file when constructing the FIELD file. It is not as difficult as it sounds!

The entry of the molecular details begins with the mandatory directive:

molecules *n*

where n is an integer specifying the number of different *types* of molecule appearing in the FIELD file. Once this directive has been encountered, DL_POLY_3 enters the *molecular description* environment in which only molecular description keywords and data are valid.

Immediately following the **molecules** directive, are the records defining individual molecules:

1. *name-of-molecule*
   which can be any character string up to 100 characters in length. (Note: this is not a directive, just a simple character string.)
2. **nummols *n***
   where n is the number of times a molecule of this type appears in the simulated system. The molecular data then follow in subsequent records:
3. **atoms *n***
   where n indicates the number of atoms in this type of molecule. A number of records follow, each giving details of the atoms in the molecule i.e. site names, masses and charges. Each record carries the entries:

`sitnam`	a8	atomic site name
`weight`	real	atomic site mass
`chge`	real	atomic site charge
`nrept`	integer	repeat counter
`ifrz`	integer	'frozen' atom (if `ifrz` $>$ 0)

The integer `nrept` need not be specified if the atom/site is not frozen (in which case a value of 1 is assumed.) A number greater than 1 specified here indicates that the next (`nrept` - 1) entries in the CONFIG file are ascribed the atomic characteristics given in the current record. The sum of the repeat numbers for all atoms in a molecule should equal the number specified by the **atoms** directive.

4. **shell *n***
   where n is the number of core-shell units. Each of the subsequent n records contains:

index 1 (i)	integer	site index of core
index 2 (j)	integer	site index of shell
spring (k)	real	force constant of core-shell spring

   The spring potential is

$$U(r) \;=\; \frac{1}{2}\; k\; r_{ij}^2 \quad , \tag{5.5}$$

   with the force constant k entered in units of `engunit` Å^{-2}, where `engunit` is the energy unit specified in the **units** directive.

   Note that the atomic site indices referred to above are indices arising from numbering each atom in the molecule from 1 to the number specified in the **atoms** directive for this molecule. This same numbering scheme should be used for all descriptions of this molecule, including the **constraints**, **pmf**, **bonds**, **angles**, **dihedrals**, **inversions** and **teth** entries described below. DL_POLY_3 will itself construct the global indices for all atoms in the systems.

   This directive (and associated data records) need not be specified if the molecule contains no core-shell units.

5. **constraints *n***
   where n is the number of constraint bonds in the molecule. Each of the following n records contains:

index 1	integer	first atomic site index
index 2	integer	second atomic site index
bondlength	real	constraint bond length

   This directive (and associated data records) need not be specified if the molecule contains no constraint bonds. See the note on the atomic indices appearing under the **shell** directive above.

6. **pmf *b***
   where b is the potential of mean force bondlength (Å). There follows the definitions of two PMF units:

   (a) **pmf unit *n1***
   where *n1* is the number of sites in the first unit. The subsequent *n1* records provide the site indices and weighting. Each record contains:

index	integer	atomic site index
weight	real	site weighting

(b) **pmf unit *n2***
where *n2* is the number of sites in the second unit. The subsequent *n2* records provide the site indices and weighting. Each record contains:

index	integer	atomic site index
weight	real	site weighting

This directive (and associated data records) need not be specified if no PMF constraints are present. See the note on the atomic indices appearing under the **shell** directive.

Note that if a site weighting is not supplied DL_POLY_3 will assume it is zero. However, DL_POLY_3 detects that all sites in a pmf unit have zero weighting then the pmf unit sites will be assigned the masses of the original atomic sites.

The pmf bondlength applies to the distance between the centres of the two pmf units. The centre, $\vec{R}_i$, of each unit is given by

$$\underline{R}_i \;=\; \frac{\sum_{j=1}^{n_i} w_j \; \vec{r}_j}{\sum_{j=1}^{n_j} w_j} \quad , \tag{5.6}$$

where r_j is a site position and w_j the site weighting.

Note that the pmf constraint is intramolecular. To define a constraint between two molecules, the molecules must be described as part of the same DL_POLY_3 "molecule". DL_POLY_3 allows only one type of pmf constraint per system. The value of nummols for this molecule determines the number of pmf constraint in the system.

Note that in DL_POLY_3 pmf constraints are handeled in every available ensemble.

7. **teth *n***
where n is the number of tethered atoms in the molecule. It is followed n records specifying the tehered sites in the molecule:

tether key	a4	potential key, see Table 5.7
index 1 (i)	integer	atomic site index
variable 1	real	potential parameter, see Table 5.7
variable 2	real	potential parameter, see Table 5.7

The meaning of these variables is given in Table 5.7.

This directive (and associated data records) need not be specified if the molecule contains no flexible chemical bonds. See the note on the atomic indices appearing under the **shell** directive above.

8. **bonds *n***
where n is the number of flexible chemical bonds in the molecule. Each of the subsequent n records contains:

bond key	a4	potential key, see Table 5.8
index 1 (i)	integer	first atomic site index in bond
index 2 (j)	integer	second atomic site index in bond
variable 1	real	potential parameter, see Table 5.8
variable 2	real	potential parameter, see Table 5.8

Table 5.7: Tethering Potentials

key	potential type	Variables (1-3)			functional form
harm	Harmonic	k			$U(r) = \frac{1}{2}\ k\ (r_i - r_i^{t=0})^2$
rhrm	Restraint	k	r_c		$U(r) = \frac{1}{2}\ k\ (r_i - r_i^{t=0})^2\ :\ \lvert r_i - r_i^{t=0}\rvert \leq r_c$ $U(r) = \frac{1}{2}\ k\ r_c^2 + k\ r_c(\lvert r_i - r_i^{t=0}\rvert - r_c)\ :\ \lvert r_i - r_i^{t=0}\rvert > r_c$
quar	Quartic	k	k'	k''	$U(r) = \frac{k}{2}\ (r_i - r_i^{t=0})^2 + \frac{k'}{3}\ (r_i - r_i^{t=0})^3$ $+\frac{k''}{4}\ (r_i - r_i^{t=0})^4$

variable 3	real	potential parameter, see Table 5.8
variable 4	real	potential parameter, see Table 5.8

The meaning of these variables is given in Table 5.8.

This directive (and associated data records) need not be specified if the molecule contains no flexible chemical bonds. See the note on the atomic indices appearing under the **shell** directive above.

9. **angles *n***
where n is the number of valence angle bonds in the molecule. Each of the n records following contains:

angle key	a4	potential key, see Table 5.9
index 1 (i)	integer	first atomic site index
index 2 (j)	integer	second atomic site index (central site)
index 3 (k)	integer	third atomic site index
variable 1	real	potential parameter, see Table 5.9
variable 2	real	potential parameter, see Table 5.9
variable 3	real	potential parameter, see Table 5.9
variable 4	real	potential parameter, see Table 5.9

The meaning of these variables is given in Table 5.9.

This directive (and associated data records) need not be specified if the molecule contains no angular terms. See the note on the atomic indices appearing under the **shell** directive above.

10. **dihedrals *n***
where n is the number of dihedral interactions present in the molecule. Each of the following n records contains:

dihedral key	a4	potential key, see Table 5.10
index 1 (i)	integer	first atomic site index
index 2 (j)	integer	second atomic site index (central site)
index 3 (k)	integer	third atomic site index

Table 5.8: Chemical Bond Potentials

key	potential type	Variables (1-4)				functional form
harm **-hrm**	Harmonic	k	r_0			$U(r) = \frac{1}{2}\ k\ (r_{ij} - r_0)^2$
mors **-mrs**	Morse	E_0	r_0	k		$U(r) = E_0\ [\{1 - \exp(-k\ (r_{ij} - r_0))\}^2 - 1]$
12-6 **-126**	12-6	A	B			$U(r) = \left(\frac{A}{r_{ij}^{12}}\right) - \left(\frac{B}{r_{ij}^{6}}\right)$
rhrm **-rhm**	Restraint	k	r_0	r_c		$U(r) = \frac{1}{2}\ k\ (r_{ij} - r_0)^2\ :\ \lvert r_{ij} - r_0\rvert \le r_c$ $U(r) = \frac{1}{2}\ k\ r_c^2 + k\ r_c(\lvert r_{ij} - r_0\rvert - r_c)\ :\ \lvert r_{ij} - r_0\rvert > r_c$
quar **-qur**	Quartic	k	r_0	k'	k''	$U(r) = \frac{k}{2}\ (r_{ij} - r_0)^2 + \frac{k'}{3}\ (r_{ij} - r_0)^3$ $+\frac{k''}{4}\ (r_{ij} - r_0)^4$
buck **-bck**	Buckingham	A	ρ	C		$U(r) = A\ \exp\left(-\frac{r_{ij}}{\rho}\right) - \frac{C}{r_{ij}^6}$
coul **-cul**	Coulomb	k				$U(r) = \frac{k}{4\ \pi\ \epsilon\ \epsilon_o}\frac{q_i q_j}{r_{ij}}$

Note: Bond potentials with a dash (-) as the first character of the keyword, do not contribute to the *excluded atoms list* (see Section 2). In this case DL_POLY_3 will also calculate the non-bonded pair potentials between the described atoms, unless these are deactivated by another potential specification.

Table 5.9: Valence Angle Potentials

key	potential type	Variables (1-4)				functional form†
harm **-hrm**	Harmonic	k	θ_0			$U(\theta) = \frac{k}{2}\ (\theta - \theta_0)^2$
quar **-qur**	Quartic	k	θ_0	k'	k''	$U(\theta) = \frac{k}{2}\ (\theta - \theta_0)^2 + \frac{k'}{3}(\theta - \theta_0)^3 + \frac{k''}{4}(\theta - \theta_0)^4$
thrm **-thm**	Truncated harmonic	k	θ_0	ρ		$U(\theta) = \frac{k}{2}\ (\theta - \theta_0)^2 \exp[-(r_{ij}^8 + r_{ik}^8)/\rho^8]$
shrm **-shm**	Screened harmonic	k	θ_0	ρ_1	ρ_2	$U(\theta) = \frac{k}{2}\ (\theta - \theta_0)^2 \exp[-(r_{ij}/\rho_1 + r_{ik}/\rho_2)]$
bvs1 **-bv1**	Screened Vessal [25]	k	θ_0	ρ_1	ρ_2	$U(\theta) = \frac{k}{8(\theta-\theta_0)^2}\left\{[(\theta_0 - \pi)^2 - (\theta - \pi)^2]^2\right\} \times \exp[-(r_{ij}/\rho_1 + r_{ik}/\rho_2)]$
bvs2 **-bv2**	Truncated Vessal [26]	k	θ_0	a	ρ	$U(\theta) = k[\theta^a(\theta - \theta_0)^2(\theta + \theta_0 - 2\pi)^2 - \frac{a}{2}\pi^{a-1}$ $(\theta - \theta_0)^2(\pi - \theta_0)^3] \exp[-(r_{ij}^8 + r_{ik}^8)/\rho^8]$
hcos **-hcs**	Harmonic Cosine	k	θ_0			$U(\theta) = \frac{k}{2}\ (\cos(\theta) - \cos(\theta_0))^2$
cos **-cos**	Cosine	A	δ	m		$U(\theta) = A\ [1 + \cos(m\theta - \delta)]$

†θ is the *i*-*j*-*k* angle

Note: valence angle potentials with a dash (-) as the first character of the keyword, do not contribute to the *excluded atoms list* (see Section 2). In this case DL_POLY_3 will calculate the non-bonded pair potentials between the described atoms.

index 4 (l)	integer	fourth atomic site index
variable 1	real	first potential parameter, see Table 5.10
variable 2	real	second potential parameter, see Table 5.10
variable 3	real	third potential parameter, see Table 5.10
variable 4	real	1-4 electrostatic interaction scale factor
variable 5	real	1-4 Van der Waals interaction scale factor
variable 6	real	fourth potential parameter, see Table 5.10
variable 7	real	fifth potential parameter, see Table 5.10

The meaning of the variables 1-3,6-7 is given in Table 5.10. The variables 4 and 5 specify the scaling factor for the 1-4 electrostatic and Van der Waals non-bonded interactions respectively.

This directive (and associated data records) need not be specified if the molecule contains no dihedral angle terms. See the note on the atomic indices appearing under the **shell** directive above.

Table 5.10: Dihedral Angle Potentials

key	potential type	Variables (1-3,6-7)			functional form‡
cos	Cosine	A	δ	m	$U(\phi) = A\ [1 + \cos(m\phi - \delta)]$
harm	Harmonic	k	ϕ_0		$U(\phi) = \frac{k}{2}\ (\phi - \phi_0)^2$
hcos	Harmonic cosine	k	ϕ_0		$U(\phi) = \frac{k}{2}\ (\cos(\phi) - \cos(\phi_0))^2$
cos3	Triple cosine	A_1	A_2	A_3	$U(\phi) = \frac{1}{2}\ \{A_1\ (1 + \cos(\phi)) + A_2\ (1 - \cos(2\phi)) + A_3\ (1 + \cos(3\phi))\}$
ryck	Ryckaert-Bellemans [29]	A			$U(\phi) = A\ \{a + b\ \cos(\phi) + c\ \cos^2(\phi) + d\ \cos^3(\phi) + e\ \cos^4(\phi) + f\ \cos^5(\phi)\}$
rbf	Fluorinated Ryckaert-Bellemans [30]	A			$U(\phi) = A\ \{a + b\ \cos(\phi) + c\ \cos^2(\phi) + d\ \cos^3(\phi) + e\ \cos^4(\phi) + f\ \cos^5(\phi)) + g\ \exp(-h(\phi - \pi)^2))\}$
opls	OPLS torsion	A_0 A_3	A_1 ϕ_0	A_2	$U(\phi) = A_0 + \frac{1}{2}\ \{A_1\ (1 + \cos(\phi - \phi_0)) + A_2\ (1 - \cos(2(\phi - \phi_0))) + A_3\ (1 + \cos(4(\phi - \phi_0)))\}$

‡ϕ is the i-j-k-l dihedral angle

11. **inversions *n***
where n is the number of inversion interactions present in the molecule. Each of the following n records contains:

inversion key	a4	potential key, see Table 5.11

index 1 (i)	integer	first atomic site index (central site)
index 2 (j)	integer	second atomic site index
index 3 (k)	integer	third atomic site index
index 4 (l)	integer	fourth atomic site index
variable 1	real	potential parameter, see Table 5.11
variable 2	real	potential parameter, see Table 5.11

The meaning of the variables 1-2 is given in Table 5.11.

This directive (and associated data records) need not be specified if the molecule contains no inversion angle terms. See the note on the atomic indices appearing under the **shell** directive above.

Table 5.11: Inversion Angle Potentials

key	potential type	Variables (1-2)		functional form‡
harm	Harmonic	k	ϕ_0	$U(\phi) = \frac{k}{2}\ (\phi - \phi_0)^2$
hcos	Harmonic cosine	k	ϕ_0	$U(\phi) = \frac{k}{2}\ (\cos(\phi) - \cos(\phi_0))^2$
plan	Planar	A		$U(\phi) = A\ [1 - \cos(\phi)]$

‡ϕ is the i-j-k-l inversion angle

12. **finish**
    This directive is entered to signal to DL_POLY_3 that the entry of the details of a molecule has been completed.

    The entries for a second molecule may now be entered, beginning with the *name-of-molecule* record and ending with the **finish** directive.

    The cycle is repeated until all the types of molecules indicated by the **molecules** directive have been entered.

The user is recommended to look at the example FIELD files in the *data* directory to see how typical FIELD files are constructed.

5.1.3.2.3 Non-bonded Interactions Non-bonded interactions are identified by atom types as opposed to specific atomic indices. The following different types of non-bonded potentials are available in DL_POLY_3 ; **vdw** - van der Waals pair, **metal** - metal, **tersoff** - Tersoff, **tbp** - three-body and **fbp** - four-body. Each of these types is specified by a specific **keyword** as described bellow.

1. **vdw *n***
   where n is the number of pair potentials to be entered. It is followed by n records, each specifying a particular pair potential in the following manner:

atmnam 1	a8	first atom type
atmnam 2	a8	second atom type
key	a4	potential key, see Table 5.12
variable 1	real	potential parameter, see Table 5.12
variable 2	real	potential parameter, see Table 5.12
variable 3	real	potential parameter, see Table 5.12
variable 4	real	potential parameter, see Table 5.12
variable 5	real	potential parameter, see Table 5.12

The variables pertaining to each potential are described in Table 5.12.

Note that any pair potential not specified in the FIELD file, will be assumed to be zero.

Table 5.12: Pair Potentials

key	potential type	Variables (1-5)					functional form
12-6	12-6	A	B				$U(r) = \left(\frac{A}{r^{12}}\right) - \left(\frac{B}{r^6}\right)$
lj	Lennard-Jones	ϵ	σ				$U(r) = 4\epsilon\left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right]$
nm	n-m	E_o	n	m	r_0		$U(r) = \frac{E_o}{(n-m)}\left[m\left(\frac{r_o}{r}\right)^n - n\left(\frac{r_o}{r}\right)^m\right]$
buck	Buckingham	A	ρ	C			$U(r) = A\ \exp\left(-\frac{r}{\rho}\right) - \frac{C}{r^6}$
bhm	Born-Huggins -Meyer	A	B	σ	C	D	$U(r) = A\ \exp[B(\sigma - r)] - \frac{C}{r^6} - \frac{D}{r^8}$
hbnd	12-10 H-bond	A	B				$U(r) = \left(\frac{A}{r^{12}}\right) - \left(\frac{B}{r^{10}}\right)$
snm	Shifted force† n-m [31]	E_o	n	m	r_0	r_c‡	$U(r) = \frac{\alpha E_o}{(n-m)}\times$ $\left[m\beta^n\left\{\left(\frac{r_o}{r}\right)^n - \left(\frac{1}{\gamma}\right)^n\right\} - n\beta^m\left\{\left(\frac{r_o}{r}\right)^m - \left(\frac{1}{\gamma}\right)^m\right\}\right]$ $+\frac{nm\alpha E_o}{(n-m)}\left(\frac{r-\gamma r_o}{\gamma r_o}\right)\left\{\left(\frac{\beta}{\gamma}\right)^n - \left(\frac{\beta}{\gamma}\right)^m\right\}$
mors	Morse	E_0	r_0	k			$U(r) = E_0[\{1 - \exp(-k(r - r_0))\}^2 - 1]$
tab	Tabulation						tabulated potential

† Note: in this formula the terms α, β and γ are compound expressions involving the variables E_o, n, m, r_0 and r_c. See Section 2.3.1 for further details.
‡ Note: r_c defaults to the general van der Waals cutoff (`rvdw` or `rcut`) if it is set to zero or not specified or not specified in the FIELD file.

2. rdf n
   where n is the number of RDF pairs to be entered. It is followed by n records, each specifying

a particular RDF pair in the following manner:

atmnam 1	a8	first atom type
atmnam 2	a8	second atom type

By default in DL_POLY_2 and DL_POLY_3 every **vdw** potential specifies a RDF pair. If the control option **rdf *f*** is specified in the CONTROL file then all pairs defined in **vdw** potentials section will also have their RDF calculated. For the calculation of RDFs of systems with force fields without **vdw** potentials some fictitious ones must then be defined. The use the **rdf *n*** option (instead of **vdw *n***) provides another way to define RDF pairs, not only neater but also memory efficient since DL_POLY_3 will not allocate (additional) potential arrays that will not be used. (This option is not available in DL_POLY_2.)

Note that **rdf** and **vdw** are not complementary - i.e. if the former is used none of the pairs defined by the latter will be considered for RDF calculations.

3. metal n
where n is the number of metal potentials to be entered. It is followed by n records, each specifying a particular metal potential in the following manner:

atmnam 1	a8	first atom type
atmnam 2	a8	second atom type
key	a4	potential key, see Table 5.13
variable 1	real	potential parameter, see Table 5.13
variable 2	real	potential parameter, see Table 5.13
variable 3	real	potential parameter, see Table 5.13
variable 4	real	potential parameter, see Table 5.13
variable 5	real	potential parameter, see Table 5.13

The variables pertaining to each potential are described in Table 5.13.

Table 5.13: Metal Potential

key	potential type	Variables (1-5)					functional form
stch	Sutton-Chen	ϵ	a	n	m	C	$U_i(r) = \epsilon\left[\frac{1}{2}\sum_{j\neq i}\left(\frac{a}{r_{ij}}\right)^n - C\sqrt{\rho_i}\right]$ $\rho_i = \sum_{j\neq i}\left(\frac{a}{r_{ij}}\right)^m$

4. tersoff n
where n is the number of specified Tersoff potentials. It is followed by $2n$ records specifying n particular Tersoff single atom type parameters and $n(n+1)/2$ records specifying cross atom type parameters in the following manner:

potential 1 : record 1

atmnam	a8	atom type
key	a4	potential key, see Table 5.14

variable 1	real	potential parameter, see Table 5.14
variable 2	real	potential parameter, see Table 5.14
variable 3	real	potential parameter, see Table 5.14
variable 4	real	potential parameter, see Table 5.14
variable 5	real	cutoff range for this potential (Å) 5.14
potential 1 : record 2		
variable 6	real	potential parameter, see Table 5.14
variable 7	real	potential parameter, see Table 5.14
variable 8	real	potential parameter, see Table 5.14
variable 9	real	potential parameter, see Table 5.14
variable 10	real	potential parameter, see Table 5.14
variable 11	real	potential parameter, see Table 5.14
...	...	...
...	...	...
potential n **: record** $2n-1$		
...	...	...
potential n **: record** $2n$		
...	...	...
cross term 1 **: record** $2n+1$		
atmnam 1	a8	first atom type
atmnam 2	a8	second atom type
variable a	real	potential parameter, see Table 5.14
variable b	real	potential parameter, see Table 5.14
...	...	...
...	...	...
cross term $n(n+1)/2$ **: record** $2n+n(n+1)/2$		
...	...	...

The variables pertaining to each potential are described in Table 5.14.

Note that the fifth variable is the range at which the particular tersoff potential is truncated. The distance is in Å.

Table 5.14: Tersoff Potential

key	potential type	Variables (1-5,6-11,a-b)						functional form
ters	Tersoff	A	a	B	b	R		Potential form
	(single)	S	β	η	c	d	h	as shown in Section
	(cross)	χ	ω					2.3.3

5. tbp n
where n is the number of three-body potentials to be entered. It is followed by n records, each specifying a particular metal potential in the following manner:

atmnam 1 (i)	a8	first atom type

atmnam 2 (j)	a8	second (central) atom type
atmnam 3 (k)	a8	third atom type
key	a4	potential key, see Table 5.15
variable 1	real	potential parameter, see Table 5.15
variable 2	real	potential parameter, see Table 5.15
variable 3	real	potential parameter, see Table 5.15
variable 4	real	potential parameter, see Table 5.15
variable 5	real	cutoff range for this potential (Å)

The variables pertaining to each potential are described in Table 5.15.

Note that the fifth variable is the range at which the three body potential is truncated. The distance is in Å, measured from the central atom.

Table 5.15: Three-body Potentials

key	potential type	Variables (1-4)				functional form†
harm	Harmonic	k	θ_0			$U(\theta) = \frac{k}{2}\ (\theta - \theta_0)^2$
thrm	Truncated harmonic	k	θ_0	ρ		$U(\theta) = \frac{k}{2}\ (\theta - \theta_0)^2\ \exp[-(r_{ij}^8 + r_{ik}^8)/\rho^8]$
shrm	Screened harmonic	k	θ_0	ρ_1	ρ_2	$U(\theta) = \frac{k}{2}\ (\theta - \theta_0)^2 \exp[-(r_{ij}/\rho_1 + r_{ik}/\rho_2)]$
bvs1	Screened Vessal [25]	k	θ_0	ρ_1	ρ_2	$U(\theta) = \frac{k}{8(\theta-\theta_0)^2}\left\{[(\theta_0 - \pi)^2 - (\theta - \pi)^2]^2\right\} \times \exp[-(r_{ij}/\rho_1 + r_{ik}/\rho_2)]$
bvs2	Truncated Vessal [26]	k	θ_0	a	ρ	$U(\theta) = k\ [\theta^a(\theta - \theta_0)^2(\theta + \theta_0 - 2\pi)^2 - \frac{a}{2}\pi^{a-1}(\theta - \theta_0)^2(\pi - \theta_0)^3]\ \exp[-(r_{ij}^8 + r_{ik}^8)/\rho^8]$
hbnd	H-bond [15]	D_{hb}	R_{hb}			$U(\theta) = D_{hb}\ \cos^4(\theta) \times [5(R_{hb}/r_{jk})^{12} - 6(R_{hb}/r_{jk})^{10}]$

†θ is the i-j-k angle

6. fbp n
where n is the number of four-body potentials to be entered. It is followed by n records, each specifying a particular metal potential in the following manner:

atmnam 1 (i)	a8	first (central) atom type
atmnam 2 (j)	a8	second atom type
atmnam 3 (k)	a8	third atom type
atmnam 3 (l)	a8	forth atom type
key	a4	potential key, see Table 5.16
variable 1	real	potential parameter, see Table 5.16
variable 2	real	potential parameter, see Table 5.16
variable 3	real	cutoff range for this potential (Å)

The variables pertaining to each potential are described in Table 5.16.

Note that the third variable is the range at which the four-body potential is truncated. The distance is in Å, measured from the central atom.

Table 5.16: Four-body Potentials

key	potential type	Variables (1-2)		functional form‡
harm	Harmonic	k	ϕ_0	$U(\phi) = \frac{k}{2}\ (\phi - \phi_0)^2$
hcos	Harmonic cosine	k	ϕ_0	$U(\phi) = \frac{k}{2}\ (\cos(\phi) - \cos(\phi_0))^2$
plan	Planar	A		$U(\phi) = A\ [1 - \cos(\phi)]$

‡ϕ is the *i-j-k-l* inversion angle

5.1.3.3 External Field

The presence of an external field is flagged by the directive:

extern

The following line in the FIELD file must have another directive indicating what type of field is to be applied. The variables pertaining to each potential are described in Table 5.17.

Note: only one type of field can be applied at a time.

5.1.3.4 Closing the FIELD File

The FIELD file must be closed with the directive:

close
which signals the end of the force field data. Without this directive DL_POLY_3 will abort.

5.1.4 The REFERENCE File

The REFERENCE has the same format and structure as CONFIG (see Section 5.1.2) file with the exception that `imcon` **MUST BE** $\neq 0$. REFERENCE may contain more or less particles than CONFIG does and may have particles whith identities that are not defined in FIELD (see Section 5.1.3). The positions of these particles are used to define the crystalline lattice sites to whitch the particles in CONFIG compare during simulation when the defect detection option, **defe**cts, is used. REFERENCE is read by the subroutine DEFECTS_REFERENCE_READ.

Table 5.17: External Fields

key	potential type	Variables (1-4)				functional form†
elec	Electric Field	E_x	E_y	E_z		$\underline{F} = q\ \underline{E}$
oshm	Oscillating Shear	A	n			$\underline{F}_x = A\ \cos(2n\pi\ z/L_z)$
shrx	Continuous Shear	A	z_0			$\underline{v}_x = \frac{A}{2}\frac{\lvert z\rvert}{z}\ :\ \lvert z\rvert\ >\ z_0$
grav	Gravitational Field	G_x	G_y	G_z		$\underline{F} = m\ \underline{G}$
magn	Magnetic Field	H_x	H_y	H_z		$\underline{F} = q\ (\underline{v} \times \underline{H})$
sphr	Containing Sphere	A	R_0	n	$R_{\rm cut}$	$\underline{F} = A\ (R_0 - r)^{-n}\ :\ r > R_{\rm cut}$
zbnd	Repulsive Wall	A	z_0	p		$\underline{F} = A\ (z_0 - z)\ :\ p\ z > p\ z_0$

5.1.5 The REVOLD File

This file contains statistics arrays from a previous job. It is not required if the current job is not a continuation of a previous run (i.e. if the **restart** directive is not present in the CONTROL file - see above). The file is unformatted and therefore not human readable. DL_POLY_3 normally produces the file REVIVE (see Section 5.2.5) at the end of a job which contains the statistics data. REVIVE should be copied to REVOLD before a continuation run commences. This may be done by the *copy* macro supplied in the *execute* sub-directory of DL_POLY_3 .

5.1.5.1 Format

The REVOLD file is unformatted. All variables appearing are written in native working precision (see Section 4.3.5) real representation. Nominally, integer quantities (e.g. the timestep number `nstep`) are represented by the the nearest real number. The contents are as follows (the dimensions of array variables are given in brackets, in terms of parameters from the SETUP_MODULE file - see Section 6.2.8).

record 1:

`nstep`	timestep of final configuration
`numacc`	number of configurations used in averages
`numrdf`	number of configurations used in rdf averages
`numzdn`	number of configurations used in Z-density averages
`time`	elapsed simulation time
`tmst`	elapsed simulation before averages were switched on
`chit`	thermostat related quantity (first)
`chip`	barostat related quantity
`cint`	thermostat related quantity (second)

record 2:

`eta`	scaling factors for simulation cell matrix elements (9)

record 3:
`stpval` instantaneous values of thermodynamic variables (`mxnstk`)
record 4:
`sumval` average values of thermodynamic variables (`mxnstk`)
record 5:
`ssqval` fluctuation (squared) of thermodynamic variables (`mxnstk`)
record 6:
`zumval` running totals of thermodynamic variables (`mxnstk`)
record 7:
`ravval` rolling averages of thermodynamic variables (`mxnstk`)
record 8:
`stkval` stacked values of thermodynamic variables (`mxstak`×`mxnstk`)
record 9:
`strcon` constraint bond stress (9)
record 10:
`strpmf` pmf constraint stress (9)
record 11:
`stress` atomic stress (9)
record 12: (Optional)
`rdf` RDF array (`mxgrdf`×`mxrdf`)
record 13: (Optional)
`zdens` Z-density array (`mxgrdf`×`mxatyp`)

5.1.5.2 Further Comments

Note that different versions of DL_POLY_3 may have a different order of the above parameters or include more or less such. Therefore a different versions of DL_POLY_3 may render any existing REVOLD file unreadable by the code.

5.1.6 The TABLE File

The TABLE file provides an alternative way of reading in the short range potentials - in tabular form. This is particularly useful if an analytical form of the potential does not exist or is too complicated to specify in the VDW_GENERATE subroutine. The table file is read by the subroutine READ_TABLE (see Chapter 6).

The option of using tabulated potentials is specified in the FIELD file (see above). The specific potentials that are to be tabulated are indicated by the use of the **tab** keyword on the record defining the short range potential (see Table 5.12).

5.1.6.1 The TABLE File Format

The file is free-formatted but blank and commented lines are not allowed.

5.1.6.2 Definitions of Variables

record 1

`header`	a100	file header
record 2		
`delpot`	real	mesh resolution in Å
`cutpot`	real	cutoff used to define tables Å
`ngrid`	integer	number of grid points in tables

The subsequent records define each tabulated potential in turn, in the order indicated by the specification in the FIELD file. Each potential is defined by a header record and a set of data records with the potential and force tables.

header record:		
`atom 1`	a8	first atom type
`atom 2`	a8	second atom type
potential data records:	(*number of data records* = `Int`((`ngrid`+3)/4))	
`data 1`	real	data item 1
`data 2`	real	data item 2
`data 3`	real	data item 3
`data 4`	real	data item 4
force data records:	(*number of data records* = `Int`((`ngrid`+3)/4))	
`data 1`	real	data item 1
`data 2`	real	data item 2
`data 3`	real	data item 3
`data 4`	real	data item 4

5.1.6.3 Further Comments

It should be noted that the number of grid points in the TABLE file should not be less than the number of grid points DL_POLY_3 is expecting. (This number is given by the parameter `mxgrid` calculated in the SETUP_MODULE file - see Section 4.2.1.3 and 6.2.8.) DL_POLY_3 will re-interpolate the tables if `ngrid`>`mxgrid`, but will abort if `ngrid`<`mxgrid`.

The potential and force tables are used to fill the internal arrays `vvv` and `ggg` respectively (see Section 2.3.1). The contents of force arrays are derived from the potential via the formula:

$$G(r) = -r\frac{\partial}{\partial r}U(r) \quad . \tag{5.7}$$

Note this is *not* the same as the true force.

5.2 The OUTPUT Files

DL_POLY_3 produces up to eight output files: HISTORY, DEFECTS, OUTPUT, REVCON, REVIVE, RDFDAT, ZDNDAT and STATIS. These respectively contain: an incremental dump file of all atomic coordinates, velocities and forces; an incremental dump file of atomic coordinates of defected particles (interstitials) and sites (vacancies); an incremental summary file of the simulation; a restart (final) configuration file; a restart (final) statistics accumulators file; a radial distribution data file; Z-density data file and a statistical history file.

5.2.1 The HISTORY File

The HISTORY file is the dump file of atomic coordinates, velocities and forces. Its principal use is for off-line analysis. The file is written by the subroutine TRAJECTORY_WRITE. The control variables for this file are `ltraj, nstraj, istraj` and `keytrj` which are created internally, based on information read from the **traj** directive in the CONTROL file (see Section 5.1.1). The HISTORY file will be created only if the directive **traj** appears in the CONTROL file.

The HISTORY file can become *very* large, especially if it is formatted. For serious simulation work it is recommended that the file be written to a scratch disk capable of accommodating a large data file. Alternatively the file may be written as unformatted (users must change that themselfs and recompile), which has the additional advantage of speed. However, writing an unformatted file has the disadvantage that the file may not be readily readable except by the machine on which it was created.

The HISTORY has the following structure:

record 1		
`header`	a72	file header
record 2		
`keytrj`	integer	trajectory key (see Table 5.1)
`imcon`	integer	periodic boundary key (see Table 5.6)
`megatm`	integer	number of atoms in simulation cell

For timesteps greater than `nstraj` the HISTORY file is appended at intervals specified by the **traj** directive in the CONTROL file, with the following information for each configuration:

record i		
`timestep`	a8	the character string "timestep"
`nstep`	integer	the current time-step
`megatm`	integer	number of atoms in simulation cell (again)
`keytrj`	integer	trajectory key (again)
`imcon`	integer	periodic boundary key (again)
`tstep`	real	integration timestep (ps)
`time`	real	elapsed simulation time (ps)
record ii		
`cell(1)`	real	x component of a cell vector
`cell(2)`	real	y component of a cell vector
`cell(3)`	real	z component of a cell vector
record iii		
`cell(4)`	real	x component of b cell vector
`cell(5)`	real	y component of b cell vector
`cell(6)`	real	z component of b cell vector
record iv		
`cell(7)`	real	x component of c cell vector
`cell(8)`	real	y component of c cell vector
`cell(9)`	real	z component of c cell vector

This is followed by the configuration for the current timestep. i.e. for each atom in the system the following data are included:

record a		
`atmnam`	a8	atomic label
`iatm`	integer	atom index
`weight`	real	atomic mass (a.m.u.)
`charge`	real	atomic charge (e)
`rsd`	real	displacement from position at $t = 0$ (Å)
record b		
`xxx`	real	x coordinate
`yyy`	real	y coordinate
`zzz`	real	z coordinate
record c only for `keytrj` > 0		
`vxx`	real	x component of velocity
`vyy`	real	y component of velocity
`vzz`	real	z component of velocity
record d only for `keytrj` > 1		
`fxx`	real	x component of force
`fyy`	real	y component of force
`fzz`	real	z component of force

Thus the data for each atom is a minimum of two records and a maximum of 4.

5.2.2 The DEFECTS File

The DEFECTS file is the dump file of atomic coordinates of defects (see Section 5.1.4). Its principal use is for off-line analysis. The file is written by the subroutine DEFECTS_WRITE. The control variables for this file are `ldef, nsdef, isdef` and `rdef` which are created internally, based on information read from the **defe** directive in the CONTROL file (see Section 5.1.1). The DEFECTS file will be created only if the directive **defe** appears in the CONTROL file.

The DEFECTS file may become *very* large, especially if it is formatted. For serious simulation work it is recommended that the file be written to a scratch disk capable of accommodating a large data file. Alternatively the file may be written as unformatted (users must change that themselfs and recompile), which has the additional advantage of speed. However, writing an unformatted file has the disadvantage that the file may not be readily readable except by the machine on which it was created.

The DEFECTS has the following structure:

record 1		
`header`	a72	file header

For timesteps greater than `nsdef` the DEFECTS file is appended at intervals specified by the **defe** directive in the CONTROL file, with the following information for each configuration:

record i		
`timestep`	a8	the character string "timestep"
`nstep`	integer	the current time-step
`tstep`	real	integration timestep (ps)
`time`	real	elapsed simulation time (ps)

`imcon`	integer	periodic boundary key (see Table 5.6)
`rdef`	real	site-interstitial cutoff (Å)
record ii		
`defects`	a7	the character string "defects"
`ndefs`	integer	the total number of defects
`interstitials`	a13	the character string "interstitials"
`ni`	integer	the total number of interstitials
`vacancies`	a9	the character string "vacancies"
`nv`	integer	the total number of vacancies
record iii		
`cell(1)`	real	x component of a cell vector
`cell(2)`	real	y component of a cell vector
`cell(3)`	real	z component of a cell vector
record iv		
`cell(4)`	real	x component of b cell vector
`cell(5)`	real	y component of b cell vector
`cell(6)`	real	z component of b cell vector
record v		
`cell(7)`	real	x component of c cell vector
`cell(8)`	real	y component of c cell vector
`cell(9)`	real	z component of c cell vector

This is followed by the `ni` interstitials for the current timestep, as each interstitial has the following data lines:

record a		
`atmnam`	a10	i_atomic label from CONFIG
`iatm`	integer	atom index from CONFIG
record b		
`xxx`	real	x coordinate
`yyy`	real	y coordinate
`zzz`	real	z coordinate

This is followed by the `nv` vacancies for the current timestep, as each vacancy has the following data lines:

record a		
`atmnam`	a10	v_atomic label from REFERENCE
`iatm`	integer	atom index from REFERENCE
record b		
`xxx`	real	x coordinate from REFERENCE
`yyy`	real	y coordinate from REFERENCE
`zzz`	real	z coordinate from REFERENCE

5.2.3 The OUTPUT File

The job output consists of 7 sections: Header; Simulation control specifications; Force field specification; System specification; Summary of the initial configuration; Simulation progress; Sample of the final configuration; Summary of statistical data; and Radial distribution functions and Z-density profile. These sections are written by different subroutines at various stages of a job. Creation of the OUTPUT file *always* results from running DL_POLY_3 . It is meant to be a human readable file, destined for hardcopy output.

5.2.3.1 Header

Gives the DL_POLY_3 version number, the number of processors in use, the link-cell algorithm in use and a title for the job as given in the header line of the input file CONTROL. This part of the file is written from the subroutines DL_POLY_, SET_BOUNDS and READ_CONTROL.

5.2.3.2 Simulation Control Specifications

Echoes the input from the CONTROL file. Some variables may be reset if illegal values were specified in the CONTROL file. This part of the file is written from the subroutine READ_CONTROL.

5.2.3.3 Force Field Specification

Echoes the FIELD file. A warning line will be printed if the system is not electrically neutral. This warning will appear immediately before the non-bonded short-range potential specifications. This part of the file is written from the subroutine READ_FIELD.

5.2.3.4 System Specification

Echoes system name, periodic boundary specification, the cell vectors and volume, some initial estimates of long-range corrections the energy and pressure (if appropriate) and degrees of freedom break-down list. This part of the file is written from the subroutines READ_CONFIG, SYSTEM_INIT and SET_TEMPERATURE.

5.2.3.5 Summary of the Initial Configuration

This part of the file is written from the main subroutine DL_POLY_. It states the initial configuration of (a maximum of) 20 atoms in the system. The configuration information given is based on the value of `levcfg` in the CONFIG file. If `levcfg` is 0 (or 1) positions (and velocities) of the 20 atoms are listed. If `levcfg` is 2 forces are also written out.

5.2.3.6 Simulation Progress

This part of the file is written by the DL_POLY_3 root segment DL_POLY_. The header line is printed at the top of each page as:

```
    step   eng_tot  temp_tot   eng_cfg   eng_src   eng_cou   eng_bnd   eng_ang   eng_dih   eng_tet
time(ps)    eng_pv  temp_rot   vir_cfg   vir_src   vir_cou   vir_bnd   vir_ang   vir_con   vir_tet
cpu  (s)    volume  temp_shl   eng_shl   vir_shl     alpha      beta     gamma   vir_pmf     press

-----------------------------------------------------------------------------------------------
```

The labels refer to :

line 1

`step`	MD step number
`eng_tot`	total internal energy of the system
`temp_tot`	system temperature
`eng_cfg`	configurational energy of the system
`eng_src`	configurational energy due to short-range potential contributions
`eng_cou`	configurational energy due to electrostatic potential
`eng_bnd`	configurational energy due to chemical bond potentials
`eng_ang`	configurational energy due to valence angle and three-body potentials
`eng_dih`	configurational energy due to dihedral inversion and four-body potentials
`eng_tet`	configurational energy due to tethering potentials

line 2

`time(ps)`	elapsed simulation time (ps) since the beginning of the job
`eng_pv`	enthalpy of system
`temp_rot`	rotational temperature
`vir_cfg`	total configurational contribution to the virial
`vir_src`	short range potential contribution to the virial
`vir_cou`	electrostatic potential contribution to the virial
`vir_bnd`	chemical bond contribution to the virial
`vir_ang`	angular and three body potentials contribution to the virial
`vir_con`	constraint bond contribution to the virial
`vir_tet`	tethering potential contribution to the virial

line 3

`cpu (s)`	elapsed cpu time since the beginning of the job
`volume`	system volume
`temp_shl`	core-shell temperature
`eng_shl`	configurational energy due to core-shell potentials
`vir_shl`	core-shell potential contribution to the virial
`alpha`	angle between b and c cell vectors
`beta`	angle between c and a cell vectors
`gamma`	angle between a and b cell vectors
`vir_pmf`	pmf constraint contribution to the virial
`press`	pressure

Note: The total internal energy of the system (variable `tot_energy`) includes all contributions to the energy (including system extensions due to thermostats etc.). It is nominally the *conserved variable* of the system, and is not to be confused with conventional system energy, which is a sum of the kinetic and configuration energies.

The interval for printing out these data is determined by the directive **print** in the CONTROL file. At each time-step that printout is requested the instantaneous values of the above statistical variables are given in the appropriate columns. Immediately below these three lines of output the

rolling averages of the same variables are also given. The maximum number of time-steps used to calculate the rolling averages is controlled by the directive **stack** in file CONTROL (see above) and listed as parameter `mxstak` in the SETUP_MODULE file (see Section 6.2.2). The default value is `mxstak` = 100.

Energy Units: The energy unit for the data appearing in the OUTPUT is defined by the **units** directive appearing in the FIELD file. System energies are therefore read in **units** per MD cell.

Pressure units: The unit of pressure is *katms*, irrespective of what energy unit is chosen.

5.2.3.7 Sample of Final Configuration

The positions, velocities and forces of the 20 atoms used for the sample of the initial configuration (see above) are given. This is written by the main subroutine DL_POLY_.

5.2.3.8 Summary of Statistical Data

This portion of the OUTPUT file is written from the subroutine STATISTICS_RESULT. The number of time-steps used in the collection of statistics is given. Then the averages over the production portion of the run are given for the variables described in the previous section. The root mean square variation in these variables follow on the next two lines. The energy and pressure units are as for the preceding section.

Also provided in this section are estimates of the diffusion coefficient and the mean square displacement for the different species in the simulation. These are determined from a *single time origin* and are therefore approximate. Accurate determinations of the diffusion coefficients can be obtained using the MSD utility program, which processes the HISTORY file (see DL_POLY_2 User Manual).

If an NPT ($N\underline{\underline{\sigma}}T$) simulation is performed the OUTPUT file also provides the mean pressure (stress tensor) and mean simulation cell vectors.

5.2.3.9 Radial Distribution Functions

If both calculation and printing of radial distribution functions have been requested (by selecting directives **rdf** and **print rdf** in the CONTROL file) radial distribution functions are printed out. This is written from the subroutine RDF_COMPUTE. First the number of time-steps used for the collection of the histograms is stated. Then each pre-requested function is given in turn. For each function a header line states the atom types ('a' and 'b') represented by the function. Then r, $g(r)$ and $n(r)$ are given in tabular form. Output is given from 2 entries before the first non-zero entry in the $g(r)$ histogram. $n(r)$ is the average number of atoms of type 'b' within a sphere of radius r around an atom of type 'a'. Note that a readable version of these data is provided by the RDFDAT file (below).

5.2.3.10 Z-density Profile

If both calculation and printing of Z-density profiles has been requested (by selecting directives **zden** and **print zden** in the CONTROL file Z-density profiles are printed out as the last part of the OUTPUT file. This is written by the subroutine Z_DENSITY_COMPUTE. First the number of

time-steps used for the collection of the histograms is stated. Then each function is given in turn. For each function a header line states the atom type represented by the function. Then z, $\rho(z)$ and $n(z)$ are given in tabular form. Output is given from $Z = [-L/2, L/2]$ where L is the length of the MD cell in the Z direction and $\rho(z)$ is the mean number density. $n(z)$ is the running integral from $-L/2$ to z of (xy cell area)$\rho(s)ds$. Note that a readable version of these data is provided by the ZDNDAT file (below).

5.2.4 The REVCON File

This file is formatted and written by the subroutine REVIVE. REVCON is the restart configuration file. The file is written every `ndump` time steps in case of a system crash during execution and at the termination of the job. A successful run of DL_POLY_3 will always produce a REVCON file, but a failed job may not produce the file if an insufficient number of timesteps have elapsed. `ndump` is controlled by the directive **dump** in file CONTROL (see above) and listed as parameter `ndump` in the SETUP_MODULE file (see Section 6.2.2). The default value is `ndump` = 1000. REVCON is identical in format to the CONFIG input file (see Section 5.1.2). REVCON should be renamed CONFIG to continue a simulation from one job to the next. This is done for you by the *copy* macro supplied in the *execute* directory of DL_POLY_3 .

5.2.5 The REVIVE File

This file is unformatted and written by the subroutine SYSTEM_REVIVE. It contains the accumulated statistical data. It is updated whenever the file REVCON is updated (see previous section). REVIVE should be renamed REVOLD to continue a simulation from one job to the next. This is done by the *copy* macro supplied in the *execute* directory of DL_POLY_3 . In addition, to continue a simulation from a previous job the **restart** keyword must be included in the CONTROL file.

The format of the REVIVE file is identical to the REVOLD file described in Section 5.1.5.

5.2.6 The RDFDAT File

This is a formatted file containing em Radial Distribution Function (RDF) data. Its contents are as follows:

record 1		
`cfgname`	a72	configuration name
record 2		
`ntprdf`	integer	number of different RDF pairs tabulated in file
`mxgrdf`	integer	number of grid points for each RDF pair

There follow the data for each individual RDF, i.e. `ntprdf` times. The data supplied are as follows:

first record		
`atname 1`	a8	first atom name
`atname 2`	a8	second atom name
following records (*mxgrdf* records)		
`radius`	real	interatomic distance (Å)
`g(r)`	real	RDF at given radius

Note the RDFDAT file is optional and appears when the **print rdf** option is specified in the CONTROL file.

5.2.7 The ZDNDAT File

This is a formatted file containing the Z-density data. Its contents are as follows:

record 1		
`cfgname`	a72	configuration name
record 2		
`ntpatm`	integer	number of unique atom types profiled in file
`mxgrdf`	integer	number of grid points in the Z-density function

There follow the data for each individual Z-density function, i.e. `ntpatm` times. The data supplied are as follows:

first record		
`atname`	a8	unique atom name
following records (*mxgrdf* records)		
`z`	real	distance in z direction (Å)
$\rho(z)$	real	Z-density at given height `z`

Note the ZDNDAT file is optional and appears when the **print rdf** option is specified in the CONTROL file.

5.2.8 The STATIS File

The file is formatted, with integers as "i10" and reals as "e14.6". It is written by the subroutine STATISTICS_COLLECT. It consists of two header records followed by many data records of statistical data.

record 1		
`cfgname`	a72	configuration name
record 2		
`string`	a8	energy units

Data records
Subsequent lines contain the instantaneous values of statistical variables dumped from the array `stpval`. A specified number of entries of `stpval` are written in the format "(1p,5e14.6)". The number of array elements required (determined by the parameter `mxnstk` in the SETUP_MODULE file) is

$$\begin{aligned} \texttt{mxnstk} \geq\ & 27 + \texttt{ntpatm}\ (\text{number of unique atomic sites})\ + \\ & 9\ (\text{stress tensor elements})\ + \\ & 9\ (\text{if constant pressure simulation requested}) \end{aligned}$$

The STATIS file is appended at intervals determined by the **stats** directive in the CONTROL file. The energy unit is as specified in the FIELD file with the **units** directive, and are compatible with the data appearing in the OUTPUT file. The contents of the appended information is:

record i		
`nstep`	integer	current MD time-step
`time`	real	elapsed simulation time
`nument`	integer	number of array elements to follow
record ii `stpval`(1) – `stpval`(5)		
`engcns`	real	total extended system energy (i.e. the conserved quantity)
`temp`	real	system temperature
`engcfg`	real	configurational energy
`engsrc`	real	short range potential energy
`engcpe`	real	electrostatic energy
record iii `stpval`(6) – `stpval`(10)		
`engbnd`	real	chemical bond energy
`engang`	real	valence angle and 3-body potential energy
`engdih`	real	dihedral, inversion, and 4-body potential energy
`engtet`	real	tethering energy
`enthal`	real	enthalpy (total energy + PV)
record iv `stpval`(11) – `stpval`(15)		
`tmprot`	real	rotational temperature
`vir`	real	total virial
`virsrc`	real	short-range virial
`vircpe`	real	electrostatic virial
`virbnd`	real	bond virial
record v `stpval`(16) – `stpval`(20)		
`virang`	real	valence angle and 3-body virial
`vircon`	real	constraint bond virial
`virtet`	real	tethering virial
`volume`	real	volume
`tmpshl`	real	core-shell temperature
record vi `stpval`(21) – `stpval`(25)		
`engshl`	real	core-shell potential energy
`virshl`	real	core-shell virial
`alpha`	real	MD cell angle α
`beta`	real	MD cell angle β
`gamma`	real	MD cell angle γ
record vii `stpval`(26) – `stpval`(27)		
`virpmf`	real	pmf constraint virial
`press`	real	pressure
the next `ntpatm` **entries**		
`amsd(1)`	real	mean squared displacement of first atom types
`amsd(2)`	real	mean squared displacement of second atom types
`...`	...	...
`amsd(ntpatm)`	real	mean squared displacement of last atom types
the next 9 entries for the stress tensor		
`stress(1)`	real	xx component of stress tensor

`stress(2)`	real	xy component of stress tensor
`stress(3)`	real	xz component of stress tensor
`stress(4)`	real	yx component of stress tensor
`...`	real	...
`stress(9)`	real	zz component of stress tensor

the next 9 entries - *if* a NPT or N$\underline{\underline{\sigma}}$T simulation is undertaken

`cell(1)`	real	x component of a cell vector
`cell(2)`	real	y component of a cell vector
`cell(3)`	real	z component of a cell vector
`cell(4)`	real	x component of b cell vector
`...`	real	...
`cell(9)`	real	z component of c cell vector

Chapter 6

DL_POLY_3 Parallelisation, Source Code Modularisation and Structure

Scope of Chapter

This chapter we discuss the DL_POLY_3 parallelisation strategy, describe the he principles used in the DL_POLY_3 modularisation of the source code and list the file structure found in the *source* subdirectory.

6.1 DL_POLY_3 Parallelisation

DL_POLY_3 is a distributed parallel molecular dynamics package based on the Domain Decomposition parallelisation strategy [2, 8, 9, 4, 5]. In this section we briefly outline the basic methodology. Users wishing to add new features DL_POLY_3 will need to be familiar with the underlying techniques as they are described in the above references.

6.1.1 The Domain Decomposition Strategy

The Domain Decomposition (DD) strategy [2, 4] is one of several ways to achieve parallelisation in MD. Its name derives from the division of the simulated system into equi-geometrical spatial blocks or domains, each of which is allocated to a specific processor of a parallel computer. I.e. the arrays defining the atomic coordinates $\underline{r}_i$, velocities $\underline{v}_i$ and forces $\underline{f}_i$, for all N atoms in the simulated system, are divided in to sub-arrays of approximate size N/P, where P is the number of processors, and allocated to specific processors. In DL_POLY_3 the domain allocation is handled by the routine DOMAINS_MODULE and the decision of approximate sizes of various bookkeeping arrays in SET_BOUNDS. The division of the configuration data in this way is based on the location of the atoms in the simulation cell, such a *geometric* allocation of system data is the hallmark of DD algorithms. Note that in order for this strategy to work efficiently, the simulated system must possess a reasonably uniform density, so that each processor is allocated an equal portion of atom data (as far as possible). Through this approach the forces computation and integration of the equations of motion are shared (reasonably) equally between processors and to a large extent can be computed independently on each processor. The method is conceptually simple though tricky to program and is particularly suited to large scale simulations, where efficiency is highest.

The DD strategy underpinning DL_POLY_3 is based on the link cell algorithm of Hockney and Eastwood [46] as implemented by various authors (e.g. Pinches *et al.* [8] and Rapaport [9]). This requires that the cutoff applied to the interatomic potentials is relatively short ranged. In DL_POLY_3 the link-cell list is build by the routine LINK_CELL_PAIRS. As with all DD algorithms, there is a need for the processors to exchange 'halo data', which in the context of link-cells means sending the contents of the link cells at the boundaries of each domain, to the neighbouring processors, so that each may have all necessary information to compute the pair forces acting on the atoms belonging to its allotted domain. This in DL_POLY_3 is handled by the SET_HALO_PARTICLES routine. three-body and four-body Systems containing complex molecules present several difficulties. They often contain ionic species, which usually require Ewald summation methods [18, 47], and *intra*-molecular interactions in addition to *inter*-molecular forces. Intramolecular interactions are handled in the same way as in DL_POLY_2, where each processor is allocated a subset of intramolecular bonds to deal with. The allocation in this case is based on the atoms present in the processor's domain. The SHAKE and RATTLE algorithms [40, 19] require significant modification. Each processor must deal with the constraint bonds present in its own domain, but it must also deal with bonds it effectively shares with its neighbouring processors. This requires each processor to inform its neighbours whenever it updates the position of a shared atom during every SHAKE (RATTLE_VV1) cycle (for RATTLE_VV2 it is velocity that needs to be updated), so that all relevant processors may incorporate this update into its own iterations. In the case of the DD strategy the SHAKE (RATTLE) algorithm is simpler than for the Replicated Data method of DL_POLY_2, where global updates of the atom positions (merging and splicing) are required [48]. The absence of the merge requirement means that the DD tailored SHAKE and RATTLE are less communications dependent and thus more efficient, particularly with large processor counts.

The DD strategy is applied to complex molecular systems as follows:

1. Using the atomic coordinates $\underline{r}_i$, each processor calculates the forces acting between the atoms in its domain - this requires additional information in the form of the halo data, which must be passed from the neighbouring processors beforehand. The forces are usually comprised of:
   (a) All common forms of non-bonded atom-atom forces
   (b) Atom-atom (site-site) coulombic forces
   (c) Sutton-Chen density dependent forces (for metals) [10, 11, 12]
   (d) Tersoff density dependent forces (for hydro-carbons) [13]
   (e) 3-body valence angle and hydrogen bond forces
   (f) 4-body inversion forces
   (g) Ion core-shell model polarisation forces
   (h) Tethering forces
   (i) Chemical bond forces
   (j) Valence angle forces
   (k) Dihedral and improper dihedral angle forces
   (l) Inversion forces
2. The computed forces are accumulated in atomic force arrays $\underline{f}_i$ independently on each processor
3. The force arrays are used to update the atomic velocities and positions of all the atoms in the domain
4. Any atom which effectively moves from one domain to another, is relocated to the neighbouring processor responsible for that domain.

It is important to note that load balancing (i.e. equal and concurrent use of all processors) is an essential requirement of the overall algorithm. In DL_POLY_3 this is accomplished quite naturally through the DD partitioning of the simulated system. Note that this will only work efficiently if the density of the system is reasonably uniform. THERE ARE NO LOAD BALANCING ALGORITHMS IN DL_POLY_3 TO COMPENSATE FOR A BAD DENSITY DISTRIBUTION.

6.1.2 Distributing the Intramolecular Bonded Terms

The intramolecular terms in DL_POLY_3 are managed through bookkeeping arrays which list all atoms (sites) involved in a particular interaction and point to the appropriate arrays of parameters that define the potential. Distribution of the forces calculations is accomplished by the following scheme:

1. Every atom (site) in the simulated system is assigned a unique 'global' index number from 1 to N;
2. Every processor maintains a list of the local indices of the the atoms in its domain. (This is the local atom list.)
3. Every processor also maintains a sorted (in ascending order) local list of global atom indices of the atoms in its domain. (This is the local sorted atom list.)

4. Every intramolecular bonded term U_{type} in the system has a unique index number i_{type}: from 1 to N_{type} where *type* represents a bond, angle, dihedral, or inversion. Also attached there with unique index numbers are core-shell units, bond constraint units, pmf constraint units and tethered atoms, their definition by site rather than by chemical type.

5. On each processor a pointer array $key_{type}(n_{type}, i_{type})$ carries the indices of the specific atoms involved in the potential term labelled i_{type} . The dimension n_{type} will be 1 if the term represents a tether, 1, 2 for a core-shell unit or a bond constraint unit or a bond, 1, 2, 3 for a valence angle and 1, 2, 3, 4 for a dihedral or an inversion, $1, .., n_{pmfunit_{1or2}} + 1$ for a pmf constraint unit.

6. Using the *key* array, each processor can identify the global indices of the atoms in the bond term and can use this in conjunction with the local sorted atoms list *and a binary search algorithm* to find the atoms in local atom list.

7. Using the local atom identity, the potential energy and force can be calculated.

Note that, at the start of a simulation DL_POLY_3 allocates individual bonded interactions to specific processors, based on the domains of the relevant atoms (DL_POLY_3 routine BUILD_BOOK_INTRA). This means that each processor does not have to handle every possible bond term to find those relevant to its domain. Also this allocation is updated as atoms move from domain to domain i.e. during the *relocation* process that follows the integration of the equations of motion (DL_POLY_3 routine RELOCATE_PARTICLES). Thus the allocation of bonded terms is effectively dynamic, changing in response to local changes.

6.1.3 Distributing the Non-bonded Terms

DL_POLY_3 calculates the non-bonded pair interactions using the link cell algorithm due to Hockney and Eastwood [46]. In this algorithm a relatively short ranged potential cutoff (r_{cut}) is assumed. The simulation cell is logically divided into so-called link cells, which have a width not less than (or equal to) the cutoff distance. It is easy to determine the identities of the atoms in each link cell. When the pair interactions are calculated it is already known that atom pairs can only interact if they are in the same link cell, or are in link cells that share a common face. Thus using the link cell 'address' of each atom, interacting pairs are located easily and efficiently via the 'link list' that identifies the atoms in each link cell. So efficient is this process that the the link list can be recreated every time step at negligible cost.

For reasons, partly historical, the link list is used to construct a Verlet neighbour list [18]. The Verlet list records the indices of all atoms within the cutoff radius (r_{cut}) of a given atom. The use of a neighbour list is not strictly necessary in the context of link-cells, but it has the advantage here of allowing a neat solution to the problem of 'excluded' pair interactions arising from the intramolecular terms (see below) or frozen atoms.

In DL_POLY_3 , the neighbour list is constructed *simultaneously* on each node, using the DD adaptation of the link cell algorithm to share the total burden of the work reasonably equally between nodes. Each node is thus responsible for a unique set of non-bonded interactions and the neighbour list is therefore different on each node.

A feature in the construction of the Verlet neighbour list for macromolecules is the concept of *excluded atoms*, which arises from the need to exclude certain atom pairs from the overall list. Which atom pairs need to be excluded is dependent on the precise nature of the force field model, but as a minimum atom pairs linked via extensible bonds or constraints and atoms (grouped in pairs) linked

via valence angles are probable candidates. The assumption behind this requirement is that atoms that are formally bonded in a chemical sense, should not participate in non-bonded interactions. (However this is not a universal requirement of all force fields .) The same considerations are needed in dealing with charged excluded atoms.

The modifications necessary to handle the excluded atoms are as follows. A distributed *excluded atoms list* is constructed by the DL_POLY_3 routine BUILD_EXCL_INTRA at the start of the simulation and is then used in conjunction with the Verlet neighbour list builder LINK_CELL_PAIRS to ensure that excluded interactions are left out of the pair force calculations. The excluded atoms list is also updated during the atom relocation process described above (DL_POLY_3 routine EXCHANGE_PARTICLES).

Once the neighbour list has been constructed, each node of the parallel computer may proceed independently to calculate the pair force contributions to the atomic forces (see routine TWO_BODY_FORCES).

The potential energy and forces arising from the non-bonded interactions, as well as metal and Tersoff interactions are calculated using interpolation tables. These are generated in the following routines: VDW_GENERATE, METAL_GENERATE and TERSOFF_GENERATE.

6.1.4 Modifications for the Ewald Sum

For systems with periodic boundary conditions DL_POLY_3 employs the Ewald Sum to calculate the coulombic interactions (see Section 2.4.5). It should be noted that DL_POLY_3 uses only the Smoothed Particle Mesh (SPME) form of the Ewald sum.

Calculation of the real space component in DL_POLY_3 employs the algorithm for the calculation of the non-bonded interactions outlined above, since the real space interactions are now short ranged (implemented in EWALD_REAL_FORCES routine).

The reciprocal space component is calculated using Fast Fourier Transform (FFT) scheme of the SMPE method [36], Section 2.4.5. The parallelisation of this scheme is entirely handled within the DL_POLY_3 by the 3D FFT routine PARALLEL_FFT, which is known as the Daresbury Advanced Fourier Transform, due to I.J. Bush [49]. This FFT distributes the SPME charge array over the processors in a manner that is completely commensurate with the distribution of the configuration data under the DD strategy. As a consequence the FFT handles all the necessary communication implicit in a distributed SPME application. The DL_POLY_3 subroutine EWALD_SPME_FORCES perfoms the bulk of the FFT operations and charge array construction, while SPME_FORCES calculates the forces.

Other routines required to calculate the Ewald sum include EWALD_EXCL_FORCES and EWALD_FROZEN_FORCES.

6.1.5 Metal Potentials

The simulation of metals (2.3.2) by DL_POLY_3 makes use of density dependent potentials of the Sutton-Chen type [10, 11, 12]. The dependence on the atomic density presents no difficulty however, as this class of potentials can be resolved into pair contributions. This permits the use of the distributed Verlet neighbour list as outlined above. DL_POLY_3 implements these potentials in various subroutines with names beginning with METAL_.

6.1.6 Tersoff, Three Body and Four Body Potentials

DL_POLY_3 can calculate Tersoff, three body and four body interactions. Although some of these interactions have similar terms to some intramolecular ones (three body to the bond angle and four body to inversion angle), these are not dealt with in the same way as the normal bonded interactions. They are generally very short ranged and are most effectively calculated using a link-cell scheme [46]. No reference is made to the Verlet neighbour list nor the excluded atoms list. It follows that atoms involved these interactions can interact via non-bonded (pair) forces and ionic forces also. Note that contributions from frozen pairs of atoms to these potentials are excluded. The calculation of the Tersoff three-body and four body terms is distributed over processors on the basis of the domain of the central atom in them. DL_POLY_3 implements these potentials in the following routines TERSOFF_FORCES, TERSOFF_GENERATE, THREE_BODY_FORCES and FOUR_BODY_FORCES.

6.1.7 Globally Summed Properties

The final stage in the DD strategy, is the global summation of different (by terms of potentials) contributions to energy, virial and sterss, which must be obtained as a global sum of the contributing terms calculated on all nodes.

The DD strategy does not require a global summation of the forces, unlike the Replicated Data method used in DL_POLY_2, which limits communication overheads and provides smooth parallelisation to large processor counts.

6.1.8 The Parallel (DD tailored) SHAKE and RATTLE Algorithms

The essentials of the DD tailored SHAKE and RATTLE algorithms (see Section 3.2) are as follows:

1. The bond constraints acting in the simulated system are allocated between the processors, based on the location (i.e. domain) of the atoms involved;
2. Each processor makes a list of the atoms bonded by constraints it must process. Entries are zero if the atom is not bonded.
3. Each processor passes a copy of the array to the neighbouring processors which manage the domains in contact with its own. The receiving processor compares the incoming list with its own and keeps a record of the shared atoms and the processors which share them.
4. In the first stage of the the algorithms, the atoms are updated through the usual Verlet algorithm, without regard to the bond constraints.
5. In the second (iterative) stage of the algorithms, each processor calculates the incremental correction vectors for the bonded atoms in its own list of bond constraints. It then sends specific correction vectors to all neighbours that share the same atoms, using the information compiled in step 3.
6. When all necessary correction vectors have been received and added the positions of the constrained atoms are corrected.
7. Steps 5 and 6 are repeated until the bond constraints are converged.
8. Finally, the change in the atom positions from the previous time step is used to calculate the atomic velocities.

The compilation of the list of constrained atoms on each processor, and the circulation of the list (items 1 - 3 above) is done at the start of the simulation, but thereafter needs only to be done every time a constraint bond atom is relocated from one processor to another. In this respect DD-SHAKE and DD-RATTLE resemble every other intramolecular term.

Since the allocation of constraints is based purely on geometric considerations, it is not practical to arrange for a strict load balancing of the DD-SHAKE algorithm. For many systems however, this deficiency has little practical impact on performance.

6.2 DL_POLY_3 Source Code

6.2.1 DL_POLY_3 Modularisation Principles

Modules in DL_POLY_3 are constructed to define parameters and variables (scalars and arrays) and/or develop methods that share much in common. The division is far from arbitrary and module interdependence is reduced to minimum. However, some dependencies exist which leads to the following division by groups in hierarchical order:

- **precision module**:: KINDS_F90

  The precision module defines the working precision
  tt wp of all real variables and parameters in DL_POLY_3 . By default it is set to 64-bit (double) precision. If the precision is changed, the user must check whether the specific platform supports it and make sure it is allowed for in the MPI implementation. If all is OK then the code must be recompiled.

- **MPI module**:: MPI_MODULE

  The MPI module implements all MPI functional calls used in DL_POLY_3 . It is only used when DL_POLY_3 is to be compiled in serial mode.

- **communication module**:: COMMS_MODULE (MPI_MODULE)

  The communication module defines MPI related parameters and develops MPI related functions and subroutines such us: initialisation and exit; global sum, maximum and minimum; node ID and number of nodes. It is dependent on KINDS_F90 and on MPI_MODULE if MPI is emulated for DL_POLY_3 compilation in serial mode. The MPI_MODULE implements all MPI functional calls used in DL_POLY_3 .

- **global parameters module**:: SETUP_MODULE

  The global parameters module holds all important global variables and parameters (see above). It is dependent on KINDS_F90

- **domains module**:: DOMAINS_MODULE

  The domains module defines DD parameters and maps the available computer resources on a DD grid. The module does not depend on previous modules but its mapping subroutine is dependent on KINDS_F90 and COMMS_MODULE.

- **parse module**:: PARSE_MODULE

  The parse module develops several methods used to deal with textual input: `get_line strip_blanks lower_case get_word word_2_real`. Depending on the method dependencies on KINDS_F90 COMMS_MODULE SETUP_MODULE DOMAINS_MODULE are found.

- **site module**:: SITE_MODULE

  The site module defines all site related arrays (FIELD) and is dependent on KINDS_F90 only. However, it also develops an allocation method that is dependent on SETUP_MODULE.

- **configuration module**:: CONFIG_MODULE

  The configuration module defines all configuration related arrays (CONFIG) and is dependent on KINDS_F90 only. However, it also develops an allocation method that is dependent on SETUP_MODULE.

- ***inter*-molecular interactions modules**:: VDW_MODULE METAL_MODULE TERSOFF_MODULE THREE_BODY_MODULE FOUR_BODY_MODULE

  The intermolecular modules define all variables and potential arrays needed for the calculation of the particular interaction in the DL_POLY_3 scope. They depend on KINDS_F90. Their allocation methods depend on SETUP_MODULE.

- ***intra*-molecular and site-related interactions modules**:: CORE_SHELL_MODULE CONSTRAINTS_MODULE PMF_MODULE TETHERS_MODULE BONDS_MODULE ANGLES_MODULE DIHEDRALS_MODULE INVERSIONS_MODULE

  These modules define all variables and potential arrays needed for the calculation of the particular interaction in the DL_POLY_3 scope. They depend on KINDS_F90. Their allocation methods depend on SETUP_MODULE.

- **external field module**:: EXTERNAL_FIELD_MODULE

  This module defines all variables and potential arrays needed for the application of an external field in the DL_POLY_3 scope. It depends on KINDS_F90 and its allocation method on SETUP_MODULE.

- **statistics module**:: STATISTICS_MODULE

  This module defines all variables and arrays needed for the statistical accountancy of a simulation in DL_POLY_3 . It depends on KINDS_F90 and its allocation method on SETUP_MODULE.

6.2.2 DL_POLY_3 File Structure

Generally, the DL_POLY_3 file structure can be divided into four groups as follows:

- **module files** in the *source* directory::

  KINDS_F90 COMMS_MODULE SETUP_MODULE DOMAINS_MODULE
  PARSE_MODULE SITE_MODULE CONFIG_MODULE
  VDW_MODULE METAL_MODULE METAL_MODULE TERSOFF_MODULE THREE_BODY_MODULE FOUR_BODY_MODULE
  CORE_SHELL_MODULE CONSTRAINTS_MODULE PMF_MODULE TETHERS_MODULE
  BONDS_MODULE ANGLES_MODULE DIHEDRALS_MODULE INVERSIONS_MODULE
  EXTERNAL_FIELD_MODULE STATISTICS_MODULE DEFECTS_MODULE

- **general files** in the *source* directory::

  WARNING ERROR
  NUMERIC_CONTAINER KINETIC_CONTAINER SPME_CONTAINER
  SCAN_FIELD SCAN_CONFIG SCAN_CONTROL SET_BOUNDS

READ_CONTROL
VDW_GENERATE VDW_TABLE_READ METAL_GENERATE
TERSOFF_GENERATE DIHEDRALS_14_CHECK READ_FIELD READ_CONFIG
TAG_LEGEND PASS_SHARED_UNITS COMPRESS_LEGEND
BUILD_BOOK_INTRA
BUILD_EXCL_INTRA
SCALE_TEMPERATURE UPDATE_SHARED_UNITS
CORE_SHELL_QUENCH CONSTRAINTS_TAGS CONSTRAINTS_QUENCH
PMF_COMS PMF_TAGS PMF_VCOMS PMF_QUENCH
SET_TEMPERATURE
VDW_LRC METAL_LRC SYSTEM_INIT
EXPORT_ATOMIC_DATA SET_HALO_PARTICLES
LINK_CELL_PAIRS
CORE_SHELL_ON_TOP.O
DEPORT_ATOMIC_DATA PMF_UNITS_SET COMPRESS_BOOK_INTRA
RELOCATE_PARTICLES
METAL_LD_COLLECT METAL_LD_EXPORT METAL_LD_SET_HALO
METAL_LD_COMPUTE
GPFA_WRAP PARALLEL_FFT EXCHANGE_GRID EWALD_SPME_FORCES
METAL_FORCES VDW_FORCES EWALD_REAL_FORCES
COUL_DDDP_FORCES COUL_CP_FORCES COUL_FSCP_FORCES
COUL_RFP_FORCES RDF_COLLECT EWALD_EXCL_FORCES
EWALD_FROZEN_FORCES TWO_BODY_FORCES
TERSOFF_FORCES THREE_BODY_FORCES FOUR_BODY_FORCES
CORE_SHELL_FORCES TETHERS_FORCES
BONDS_FORCES ANGLES_FORCES DIHEDRALS_FORCES INVERSIONS_FORCES
CORE_SHELL_RELAX LANGEVIN_FORCES
NVT_E_SCL NVT_B_SCL
XSCALE CORE_SHELL_KINETIC TRAJECTORY_WRITE
DEFECTS_REFERENCE_READ DEFECTS_REFERENCE_EXPORT
DEFECTS_REFERENCE_SET_HALO DEFECTS_LINK_CELLS DEFECTS_WRITE
STATISTICS_COLLECT Z_DENSITY_COLLECT SYSTEM_REVIVE
RDF_COMPUTE Z_DENSITY_COMPUTE STATISTICS_RESULT
DL_POLY

- **VV specific** files in the *source/VV* directory::

EXTERNAL_FIELD_APPLY_VV EXTERNAL_FIELD_CORRECT
PSEUDO_VV ZERO_K_OPTIMISE_VV
CONSTRAINTS_SHAKE_VV PMF_SHAKE_VV
CONSTRAINTS_RATTLE PMF_RATTLE
NVT_H_SCL NPT_H_SCL NST_H_SCL
NVE_1_VV NVT_L1_VV
NVT_E1_VV NVT_B1_VV NVT_H1_VV
NPT_B1_VV NPT_H1_VV NPT_M1_VV
NST_B1_VV NST_H1_VV NST_M1_VV
MD_VV

- **LFV specific** files in the *source/LFV* directory::

EXTERNAL_FIELD_APPLY_LFV
PSEUDO_LFV ZERO_K_OPTIMISE_LFV

CONSTRAINTS_SHAKE_LFV PMF_SHAKE_LFV
NVE_1_LFV NVT_L1_LFV
NVT_E1_LFV NVT_B1_LFV NVT_H1_LFV
NPT_B1_LFV NPT_H1_LFV NPT_M1_LFV
NST_B1_LFV NST_H1_LFV NST_M1_LFV
MD_LFV

- **SERIAL specific** files in the *source/SERIAL* directory::

  MPIF.H MPI_MODULE SET_BOUND~ EWALD_SPME_FORC~S

The files in each group are listed in hierarchial order as closely as possible. The further down the list the file, the more dependent it is on the files listed before it.

6.2.3 Module Files

The DL_POLY_3 module files contain all global variables (scalars and arrays) and parameters as well as some general methods and generic functions intrinsically related to the purpose or/and contents of the specific module. The file-names and the methods or/and functions developed in them have self-explanatory names. More information of their purpose can be found in their headers.

The rest of files in DL_POLY_3 are dependent on the module files in various ways. The dependency relation to a module file is explicitly stated in the declaration part of the code.

6.2.4 General Files

The DL_POLY_3 general files are common to both MPI and SERIAL version of the code. In most cases, they have self-explanatory names as their order is matched as closely as possible to that occurring in the main segment of the code - DL_POLY. Only the first five files are exception of that rule; WARNING and ERROR are important reporting subroutines that have call points at various places in the code, and NUMERIC_CONTAINER, KINETIC_CONTAINER and SPME_CONTAINER are containers of simple functions and subroutines related in some way to their purpose in the code.

Only one explicit internal dependence between files in this group exists: EWALD_SPME_FORCES depends on PARALLEL_FFT.

6.2.5 VV and LFV Specific Files

These implement the specific integration scheme as file-names are finished with the flavour they develop if they have a counterpart implementing the same algorithm but in the alternative flavour. Names are self-explanatory.

6.2.6 SERIAL Specific Files

These implement an emulation of some general MPI calls used in DL_POLY_3 source code when compiling in serial mode as well as some modified counterparts of the general files changed to allow for faster and/or better memory optimised serial execution. Names are self-explanatory.

6.2.7 Comments on MPI Handling

Only a few files make explicit calls to MPI routines:
COMMS_MODULE READ_CONFIG
PASS_SHARED_UNITS UPDATE_SHARED_UNITS
EXPORT_ATOMIC_DATA DEPORT_ATOMIC_DATA
METAL_LD_EXPORT PARALLEL_FFT EXCHANGE_GRID
DEFECTS_REFERENCE_READ DEFECTS_REFERENCE_EXPORT DEFECTS_WRITE
TRAJECTORY_WRITE SYSTEM_REVIVE.
The rest of the files that use MPI functionality in any way make implicit calls via generic functions developed in COMMS_MODULE.

6.2.8 Comments on SETUP_MODULE

The most important module, by far, is SETUP_MODULE, which holds the most important global parameters and variables (some of which serve as "parameters" for global array bounds, set in SET_BOUNDS). A brief account of these is given below:

parameter	**value**	**function**
`pi`	3.1415926535897932	π constant
`sqrpi`	1.7724538509055160	$\sqrt[2]{\pi}$ constant
`rt2`	1.4142135662373095	$\sqrt[2]{2}$ constant
`rt3`	1.7320508075688772	$\sqrt[2]{3}$ constant
`r4pie0`	138935.4835	electrostatics conversion factor to internal units, i.e. $\frac{1}{4\pi\epsilon_o}$
`boltz`	0.831451115	Boltzmann constant in internal units
`prsunt`	0.163882576	conversion factor for pressure from internal units to katms
`nread`	5	main input channel
`nrite`	6	main output channel
`nfield`	7	force field input channel
`nconf`	10	configuration file input channel
`nstats`	20	statistical data file output channel
`nhist`	21	trajectory history file channel
`nrest`	22	output channel accumulators restart dump file
`ntable`	23	tabulated potentials file input channel
`nrdfdt`	24	output channel for RDF data
`nzdfdt`	25	output channel for Z-density data file
`nrefdt`	26	reference configuration input channel
`ndefdt`	27	output channel for defects data file
`mxsite`	*variable*	max number of molecular sites
`mxatyp`	*variable*	max number of unique atomic types
`mxtmls`	*variable*	max number of unique molecule types
`mxspl`	*variable*	SPME FFT B-spline order
`kmaxa`	*variable*	SPME FFT array dimension (a direction)
`kmaxb`	*variable*	SPME FFT array dimension (b direction)
`kmaxc`	*variable*	SPME FFT array dimension (c direction)
`mxtshl`	*variable*	max number of specified core-shell unit types in system
`mxshl`	*variable*	max number of core-shell units per node
`mxtcon`	*variable*	max number of specified bond constraints in system

`mxcons`	*variable*	max number of constraint bonds per a node
`mxtpmf(1:2)`	*variable*	max number of specified particles in pmf units (1:2)
`mxpmf`	*variable*	max number of pmf constraints per a node
`mxlshp`	*variable*	max number of shared particles per node `Max`$(1, 2$ `mxshl`$, 2$ `mxcons`$)$
`mxproc`	*variable*	number of neighbour nodes in DD hypercube (26)
`mxtteth`	*variable*	max number of specified tethered potentials in system
`mxteth`	*variable*	max number of tethered atoms per node
`mxpteth`	*variable*	max number of parameters for tethered potentials (3)
`mxtbnd`	*variable*	max number of specified chemical bond potentials in system
`mxbond`	*variable*	max number of chemical bonds per node
`mxpbnd`	*variable*	max number of parameters for chemical bond potentials (4)
`mxtang`	*variable*	max number of specified bond angle potentials in system
`mxangl`	*variable*	max number of bond angles per node
`mxpang`	*variable*	max number of parameters for bond angle potentials (4)
`mxtdih`	*variable*	max number of specified dihedral angle potentials in system
`mxdihd`	*variable*	max number of dihedral angles per node
`mxpdih`	*variable*	max number of parameters for dihedral angle potentials (7)
`mxtinv`	*variable*	max number of specified inversion angle potentials in system
`mxinv`	*variable*	max number of inversion angles per node
`mxpinv`	*variable*	max number of parameters for inversion angle potentials (2)
`mxexcl`	*variable*	max number of excluded interactions per atom
`mxfix`	*variable*	max number of bonds links to an atom
`mxgrid`	*variable*	max number of grid points in potential arrays
`mxrdf`	*variable*	max number of pairwise RDF in system
`mxgrdf`	*variable*	number of grid points for RDF and Z-density
`mxvdw`	*variable*	max number of van der Waals potentials in system
`mxpvdw`	*variable*	max number of van der Waals potential parameters (5)
`mxmet`	*variable*	max number of metal potentials in system
`mxpmet`	*variable*	max number of metal potential parameters (5)
`mxter`	*variable*	max number of Tersoff potentials in system
`mxpter`	*variable*	max number of Tersoff potential parameters (11)
`mxtbp`	*variable*	max number of three-body potentials in system
`mx2tbp`	*variable*	array dimension of three-body potential parameters
`mxptbp`	*variable*	max number of three-body potential parameters (4)
`mxfbp`	*variable*	max number of four-body potentials in system
`mx2fbp`	*variable*	array dimension of four-body potential parameters
`mxpfbp`	*variable*	max number of four-body potential parameters (3)
`mxpfld`	*variable*	max number of external field parameters (5)
`mxstak`	*variable*	dimension of stack arrays for rolling averages
`mxnstk`	*variable*	max number of stacked variables
`mxlist`	*variable*	max number of atoms in the verlet list on a node
`mxcell`	*variable*	max number of link cells per node
`mxatms`	*variable*	max number of atoms per node
`mxbuff`	*variable*	max dimension of the principle transfer buffer

Chapter 7

Examples

Scope of Chapter

This chapter describes the standard test cases for DL_POLY_3 , the input and output files for which are in the *data* sub-directory.

7.1 Test Cases

The following example data sets (both input and output) are stored in the subdirectory *data*. These are provided so that you may check that your version of DL_POLY_3 is working correctly. All the jobs are of a size suitable to test the code in parallel execution. They are not suitable for a single processor computer. The files are stored in compressed format. The test cases can be run by typing

select n

from the *execute* directory, where n is the number of the test case. The *select* macro will copy the appropriate CONTROL, CONFIG, and FIELD files to the *execute* directory ready for execution. The output file OUTPUT may be compared with the file supplied in the *data* directory.

The example output provided in the *data* directory were obtained on an IBM SP4 parallel system.

It should be noted that the potentials and the simulation conditions used in the following test cases are chosen to demonstrate functionality only. **They are not necessarily appropriate for serious simulation of the test systems.**

7.1.1 Test Case 1 and 2: Sodium Chloride

These are a 27,000 and 216,000 ion systems respectively with unit electric charges on sodium and chlorine. Simulation at 500 K with a NVT Berendsen ensemble. The SPME method is used to calculate the Coulombic interactions.

7.1.2 Test Case 3 and 4: DPMC in Water

These systems consist of 200 and 1,600 DMPC molecules in 9379 and 75032 water molecules respectively. Simulation at 300 K using NVE ensemble with SPME and RATTLE algorithm for the constrained motion. Total system size is 51737 and 413896 atoms respectively.

7.1.3 Test Case 5 and 6: $KNaSi_2O_5$

Potassium Sodium disilicate glass ($NaKSi_2O_5$) using two and three body potentials. Some of the two body potentials are read from the TABLE file. Simulation at 1000 K using NVT Hoover ensemble with SPME. Cubic periodic boundaries are in use. System size is 69120 and 552960 ions respectively.

7.1.4 Test Case 7 and 8: Gramicidin A molecules in Water

These systems consist of 8 and 16 gramicidin A molecules in aqueous solution (32,096 and 256,768 water molecules) with total number of atoms 99,120 and 792,960 respectively. Simulation at 300 K using NPT Berendsen ensemble with SPME and SHAKE/RATTLE algorithm for the constrained motion.

7.1.5 Test Case 9 and 10: SiC with Tersoff Potentials

These systems consist of 74,088 and 343,000 atoms respectively. Simulation at 300 K using NPT Hoover ensemble with Tersoff forces and no electrostatics.

7.1.6 Test Case 11 and 12: Cu_3Au with Metal Potentials

These systems consist of 32,000 and 256,000 atoms respectively. Simulation at 300 K using NVT Hoover ensemble with metal forces and no electrostatics.

7.1.7 Test Case 13 and 14: lipid bilayer in water

These systems consist of 12,428 and 111,852 atoms respectively. Simulation at 300 K using NVT Berendsen ensemble with SPME and SHAKE/RATTLE algorithm for the constrained motion.

7.1.8 Test Case 15 and 16: relaxed and adiabatic shell model MgO

These systems consist of 8,000 (4,000 shells) and 64,000 (32,000 shells) atoms respectively. Simulation at 3000 K using NPT Berendsen ensemble with SPME. FIELD and CONTROL files for each shell model are provided separately.

7.1.9 Test Case 17 and 18: Potential of mean force on K+ in water MgO

These systems consist of 13,500 (500 PMFs) and 53,248 (2,048 PMFs) atoms respectively. Simulation at 300 K using NPT Berendsen ensemble with SPME and SHAKE/RATTLE algorithm for the constrained motion.

7.2 Benchmark Cases

DL_POLY_3 benchmark test cases are distributed only with the authors' permission.

Appendix A

DL_POLY_3 Periodic Boundary Conditions

Introduction

DL_POLY_3 is designed to accommodate a number of different periodic boundary conditions, which are defined by the shape and size of the simulation cell. Briefly, these are as follows (which also indicates the IMCON flag defining the simulation cell type in the CONFIG file - see 5.1.2):

1. None e.g. isolated polymer in space (`imcon` = 0)
2. Cubic periodic boundaries (`imcon` = 1)
3. Orthorhombic periodic boundaries (`imcon` = 2)
4. Parallelepiped periodic boundaries (`imcon` = 3)
5. Slab (X,Y periodic; Z non-periodic) (`imcon` = 6)

We shall now look at each of these in more detail. Note that in all cases the cell vectors and the positions of the atoms in the cell are to be specified in Angstroms (Å).

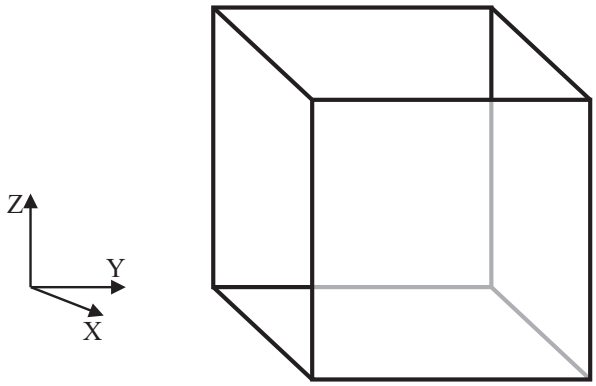
No periodic boundary (`imcon` = 0)

Simulations requiring no periodic boundaries are best suited to *in vacuuo* simulations, such as the conformational study of an isolated polymer molecule. This boundary condition is not recommended for studies in a solvent, since evaporation is likely to be a problem.

Note this boundary condition have to be used with caution. DL_POLY_3 is not naturally suited to carry out efficient calculations on systems with great fluctuation of the local density in space, as is the case for clusters in vacuum. The parallelisation and domain decomposition is therefore limited to eight domains (maximum of two in each direction in space).

This boundary condition should not used with the SPME Ewald summation method.

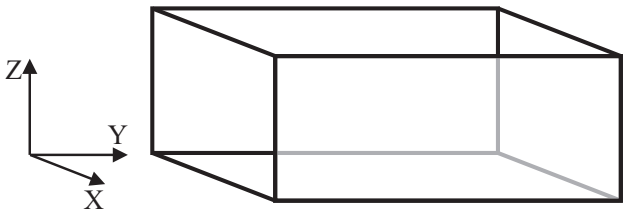
Cubic periodic boundaries (`imcon = 1`)



The cubic MD cell.

The cubic MD cell is perhaps the most commonly used in simulation and has the advantage of great simplicity. In DL_POLY_3 the cell is defined with the principle axes passing through the centres of the faces. Thus for a cube with sidelength D, the cell vectors appearing in the CONFIG file should be: (D,0,0); (0,D,0); (0,0,D). Note the origin of the atomic coordinates is the centre of the cell.

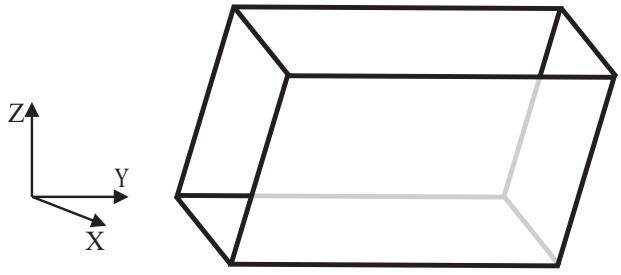
Orthorhombic periodic boundaries (`imcon = 2`)



The orthorhomic MD cell.

The orthorhombic cell is also a common periodic boundary, which closely resembles the cubic cell in use. In DL_POLY_3 the cell is defined with principle axes passing through the centres of the faces. For an orthorhombic cell with sidelengths D (in X-direction), E (in Y-direction) and F (in Z-direction), the cell vectors appearing in the CONFIG file should be: (D,0,0); (0,E,0); (0,0,F). Note the origin of the atomic coordinates is the centre of the cell.

Parallelepiped periodic boundaries (`imcon = 3`)



The parallelepiped MD cell.

The parallelepiped (e.g. monoclinic or triclinic) cell is generally used in simulations of crystalline materials, where its shape and dimension is commensurate with the unit cell of the crystal. Thus for a unit cell specified by three principal vectors $\underline{a}$, $\underline{b}$, $\underline{c}$, the MD cell is defined in the DL_POLY_3 CONFIG file by the vectors (La_1,La_2,La_3), (Mb_1,Mb_2,Mb_3), (Nc_1,Mc_2,Nc_3), in which L,M,N are integers, reflecting the multiplication of the unit cell in each principal direction. Note that the atomic coordinate origin is the centre of the MD cell.

Slab boundary conditions (`imcon` = 6)

Slab boundaries are periodic in the X- and Y-directions, but not in the Z-direction. They are particularly useful for simulating surfaces. The periodic cell in the XY plane can be any parallelogram. The origin of the X,Y atomic coordinates lies on an axis perpendicular to the centre of the parallelogram. The origin of the Z coordinate is where the user specifies it. However, it is recommended that it is in the middle of the slab. Domain decomposition division across Z axis is limited to 2.

If the XY parallelogram is defined by vectors $\underline{A}$ and $\underline{B}$, the vectors required in the CONFIG file are: (A_1,A_2,0), (B_1,B_2,0), (0,0,D), where D is any real number (including zero). If D is nonzero, it will be used by DL_POLY to help determine a 'working volume' for the system. This is needed to help calculate RDFs etc. (The working value of D is in fact taken as one of: 3×cutoff; or 2×max abs(Z coordinate)+cutoff; or the user specified D, whichever is the larger.)

The surface in a system with charges can also be modelled with DL_POLY_3 if periodicity is allowed in the Z-direction. In this case slabs of ions well-separated by vacuum zones in the Z-direction can be handled with `imcon` = 1, 2 or 3.

Appendix B

DL_POLY_3 Macros

Introduction

Macros are simple executable files containing standard unix commands. A number of the are supplied with DL_POLY_3 and are found in the *execute* sub-directory. The available macros are as follows:

- *cleanup*
- *copy*
- *gopoly*
- *gui*
- *select*
- *store*

/noindent The function of each of these is described below. It is worth noting that most of these functions can be performed by the DL_POLY Java GUI [17].

cleanup

cleanup removes several standard data files from the *execute* sub-directory. It contains the unix commands:

```
rm OUTPUT REVCON REVOLD STATIS REVIVE DEFECTS gopoly.*
```

and removes the files OUTPUT, REVCON, REVOLD, STATIS, REVIVE, DEFECTS and gopoly.* (all variants). It is useful for cleaning the sub-directory up after a run. (Useful data should be stored elsewhere however!)

copy

copy invokes the unix commands:

```
mv CONFIG CONFIG.OLD
mv REVCON CONFIG
mv REVIVE REVOLD
```

which collectively prepare the DL_POLY_3 files in the *execute* sub-directory for the continuation of a simulation. It is always a good idea to store these files elsewhere in addition to using this macro.

gopoly

gopoly is used to submit a DL_POLY_3 job to the HPCx, which operates a LOAD-LEVELER job queuing system. It invokes the following script:

```
#@ shell = /usr/bin/tcsh
#
#@ job_type = parallel
#@ job_name = gopoly
#
#@ cpus = 32
#
#@ node_usage = not_shared
#@ network.MPI = csss,shared,US
#
#@ wall_clock_limit = 00:30:00
#@ account_no = my_account
#
#@ output = $(job_name).$(schedd_host).$(jobid).out
#@ error  = $(job_name).$(schedd_host).$(jobid).err
#@ notification = never
#
#@ bulkxfer = yes
#@ data_limit = 850000000
#@ stack_limit = 10000000
#
#@ queue
#
# ENVIRONMENT SETTINGS
#
setenv MP_EAGER_LIMIT 65536
setenv MP_SHARED_MEMORY yes
setenv MEMORY_AFFINITY MCM
setenv MP_TASK_AFFINITY MCM
setenv MP_SINGLE_THREAD yes
#
poe  ./DLPOLY.Y
```

Using LOADLEVELLER, the job is submitted by the unix command:

llsubmit gopoly

where *llsubmit* is a local command for submission to the IBM SP4 cluster. The number of required nodes and the job time are indicated in the above script.

gui

gui is a macro that starts up the DL_POLY_3 Java GUI. It invokes the following unix commands:

```
java -jar ../java/GUI.jar
```

In other words the macro invokes the Java Virtual Machine which executes the instructions in the Java archive file GUI.jar, which is stored in the *java* subdirectory of DL_POLY_3 . (Note: Java 1.3.0 or a higher version is required to run the GUI.)

select

select is a macro enabling easy selection of one of the test cases. It invokes the unix commands:

```
cp ../data/TEST$1/CONTROL   CONTROL
cp ../data/TEST$1/FIELD     FIELD
cp ../data/TEST$1/CONFIG    CONFIG
cp ../data/TEST$1/TABLE     TABLE
cp ../data/TEST$1/REFERENCE REFERENCE
```

select requires one argument (an integer) to be specified:

select n

where *n* is test case number, which ranges from 1 to 18.

This macro sets up the required input files in the *execute* sub-directory to run the *n*-th test case. The command to copy the TABLE file will be unnecessary in most cases.

store

The *store* macro provides a convenient way of moving data back from the *execute* sub-directory to the *data* sub-directory. It invokes the unix commands:

```
mkdir      ../data/TEST$1
mv OUTPUT  ../data/TEST$1/OUTPUT
mv REVCON  ../data/TEST$1/REVCON
mv STATIS  ../data/TEST$1/STATIS
mv REVIVE  ../data/TEST$1/REVIVE
mv HISTORY ../data/TEST$1/HISTORY
mv DEFECTS ../data/TEST$1/DEFECTS
mv RDFDAT  ../data/TEST$1/RDFDAT
mv ZDNDAT  ../data/TEST$1/ZDNDAT
chmod 400  ../data/TEST$1/*
```

which first creates a new DL_POLY *data/TEST..* sub-directory and then moves the standard DL_POLY_3 output data files into it.

store requires one argument:

store n

where *n* is a unique string or number to label the output data in the *data/TESTn* sub-directory.

Note that *store* sets the file access to read-only. This is to prevent the *store* macro overwriting existing data without your knowledge.

Appendix C

DL_POLY_3 Makefiles

Makefile_MPI

```
# Master makefile for DL_POLY_3.06 (parallel version)
#
# Author - I.T.Todorov February 2006
#
#
# Define default settings
#====================================================================

SHELL=/bin/sh

.SUFFIXES:
.SUFFIXES: .f90 .o

BINROOT=../execute
EX=DLPOLY.Y
EXE=$(BINROOT)/$(EX)

TYPE=master

FC=undefined
LD=undefined

# Define object files
#====================================================================

OBJ_MOD = \
    kinds_f90.o comms_module.o setup_module.o \
    domains_module.o parse_module.o \
    site_module.o config_module.o defects_module.o \
    vdw_module.o metal_module.o tersoff_module.o \
    three_body_module.o four_body_module.o \
    core_shell_module.o constraints_module.o pmf_module.o tethers_module.o \
    bonds_module.o angles_module.o dihedrals_module.o inversions_module.o \
```

```
    external_field_module.o statistics_module.o

OBJ_ALL = \
    warning.o error.o \
    numeric_container.o kinetic_container.o spme_container.o \
    scan_field.o scan_config.o scan_control.o set_bounds.o \
    read_control.o \
    vdw_generate.o vdw_table_read.o metal_generate.o \
    tersoff_generate.o dihedrals_14_check.o read_field.o read_config.o \
    tag_legend.o pass_shared_units.o compress_legend.o build_book_intra.o \
    build_excl_intra.o \
    scale_temperature.o update_shared_units.o \
    core_shell_quench.o constraints_tags.o constraints_quench.o \
    pmf_coms.o pmf_tags.o pmf_vcoms.o pmf_quench.o \
    set_temperature.o \
    vdw_lrc.o metal_lrc.o system_init.o \
    export_atomic_data.o set_halo_particles.o \
    link_cell_pairs.o \
    core_shell_on_top.o \
    deport_atomic_data.o pmf_units_set.o compress_book_intra.o \
    relocate_particles.o \
    metal_ld_collect.o metal_ld_export.o metal_ld_set_halo.o \
    metal_ld_compute.o \
    gpfa_wrap.o parallel_fft.o exchange_grid.o ewald_spme_forces.o \
    metal_forces.o vdw_forces.o ewald_real_forces.o \
    coul_dddp_forces.o coul_cp_forces.o coul_fscp_forces.o \
    coul_rfp_forces.o rdf_collect.o ewald_excl_forces.o \
    ewald_frozen_forces.o two_body_forces.o \
    tersoff_forces.o three_body_forces.o four_body_forces.o \
    core_shell_forces.o tethers_forces.o \
    bonds_forces.o angles_forces.o dihedrals_forces.o inversions_forces.o \
    core_shell_relax.o langevin_forces.o \
    nvt_e_scl.o nvt_b_scl.o \
    \
    external_field_apply_vv.o external_field_correct.o \
    pseudo_vv.o zero_k_optimise_vv.o \
    constraints_shake_vv.o pmf_shake_vv.o \
    constraints_rattle.o pmf_rattle.o \
    nvt_h_scl.o npt_h_scl.o nst_h_scl.o \
    nvt_l1_vv.o nve_1_vv.o \
    nvt_e1_vv.o nvt_b1_vv.o nvt_h1_vv.o \
    npt_b1_vv.o npt_h1_vv.o npt_m1_vv.o \
    nst_b1_vv.o nst_h1_vv.o nst_m1_vv.o \
    \
    external_field_apply_lfv.o \
    pseudo_lfv.o zero_k_optimise_lfv.o \
    constraints_shake_lfv.o pmf_shake_lfv.o \
    nvt_l1_lfv.o nve_1_lfv.o \
    nvt_e1_lfv.o nvt_b1_lfv.o nvt_h1_lfv.o \
```

```
    npt_b1_lfv.o npt_h1_lfv.o npt_m1_lfv.o \
    nst_b1_lfv.o nst_h1_lfv.o nst_m1_lfv.o \
    \
    xscale.o core_shell_kinetic.o trajectory_write.o \
    defects_reference_read.o defects_reference_export.o \
    defects_reference_set_halo.o defects_link_cells.o defects_write.o \
    statistics_collect.o z_density_collect.o system_revive.o \
    rdf_compute.o z_density_compute.o statistics_result.o \
    dl_poly.o

# Define Velocity Verlet files
#=====================================================================

FILES_VV = \
    external_field_apply_vv.f90 external_field_correct.f90 \
    pseudo_vv.f90 zero_k_optimise_vv.f90 \
    constraints_shake_vv.f90 pmf_shake_vv.f90 \
    constraints_rattle.f90 pmf_rattle.f90 \
    nvt_h_scl.f90 npt_h_scl.f90 nst_h_scl.f90 \
    nve_1_vv.f90  nvt_l1_vv.f90 \
    nvt_e1_vv.f90 nvt_b1_vv.f90 nvt_h1_vv.f90 \
    npt_b1_vv.f90 npt_h1_vv.f90 npt_m1_vv.f90 \
    nst_b1_vv.f90 nst_h1_vv.f90 nst_m1_vv.f90 \
    md_vv.f90

# Define LeapFrog Verlet files
#=====================================================================

FILES_LFV = \
    external_field_apply_lfv.f90 \
    pseudo_lfv.f90 zero_k_optimise_lfv.f90 \
    constraints_shake_lfv.f90 pmf_shake_lfv.f90 \
    nve_1_lfv.f90 nvt_l1_lfv.f90 \
    nvt_e1_lfv.f90 nvt_b1_lfv.f90 nvt_h1_lfv.f90 \
    npt_b1_lfv.f90 npt_h1_lfv.f90 npt_m1_lfv.f90 \
    nst_b1_lfv.f90 nst_h1_lfv.f90 nst_m1_lfv.f90 \
    md_lfv.f90

# Examine targets manually
#=====================================================================

all:
    @echo
    @echo "You MUST specify a target machine!"
    @echo
    @echo "Please examine Makefile for permissible targets!"
    @echo
    @echo "If no target suits your system create your own"
    @echo "using the generic target template provided in"
```

```
    @echo "this Makefile at entry 'uknown_machine:'."
    @echo

# Fetch the Velocity Verlet subroutines
#=====================================================================

$(FILES_VV):
    make links_vv

links_vv:
    @for file in ${FILES_VV} ; do \
    echo linking to $$file ;\
    rm -f $$file ;\
    ln -s VV/$$file $$file ;\
    done

# Fetch the LeapFrog Verlet subroutines
#=====================================================================

$(FILES_LFV):
    make links_lfv

links_lfv:
    @for file in ${FILES_LFV} ; do \
    echo linking to $$file ;\
    rm -f $$file ;\
    ln -s LFV/$$file $$file ;\
    done

# Clean up the source directory
#=====================================================================

clean:
    rm -f $(OBJ_MOD) $(OBJ_ALL) $(FILES_VV) $(FILES_LFV) *.mod

# Generic target template
#=====================================================================
uknown_machine:
    $(MAKE) LD="path to Fortran 90 Linker-loaDer" \
    LDFLAGS="appropriate flags for LD (MPI libraries)" \
    FC="path to Fortran 90 compiler" \
    FCFLAGS="appropriate flags for FC (MPI include)" \
    EX=$(EX) BINROOT=$(BINROOT) $(TYPE)

# System specific targets follow:
#=====================================================================

#================== Beowulf Linux ifort + mpich ======================
lake:
```

```
    $(MAKE) LD="/opt/intel/compiler70/ia32/bin/ifc -v -o" \
    LDFLAGS="-O3 -xW -prec_div -L/opt/mpich-intel/lib -lmpich \
    -L/opt/intel/compiler70/ia32/lib/ -lPEPCF90" \
    FC="/opt/intel/compiler70/ia32/bin/ifc -c" \
    FCFLAGS="-O3 -xW -prec_div -I/opt/mpich-intel/include" \
    EX=$(EX) BINROOT=$(BINROOT) $(TYPE)

#================== Linux efc SGI ALTIX + parallel FFT  ==============
newton:
    $(MAKE) LD="ifort -o" LDFLAGS="-tpp2 -ip -O3 -lmpi -lguide" \
    FC="ifort -c" FCFLAGS="-O3 -tpp2 -ip -w" \
    EX=$(EX) BINROOT=$(BINROOT) $(TYPE)

#================== Beowulf Linux pgf90 + myrinet / mpich ============
dirac:
    $(MAKE) LD="/usr/local/mpich-gm-pgroup121-7b/bin/mpif90 -v -o" \
    LDFLAGS="-O3 -L/usr/local/mpich-gm-pgroup121-7b/lib -lmpich -lfmpich \
             -lmpichf90 -L/usr/local/gm/binary/lib -lgm -L/usr/local/lib" \
    FC="/usr/local/mpich-gm-pgroup121-7b/bin/mpif90 -c" \
    FCFLAGS="-fast -Knoieee -Mdalign -O3" \
    EX=$(EX) BINROOT=$(BINROOT) $(TYPE)

#================== Franklin (SUNfire cluster) ========================
#setenv HPCF_MPI yes
franklin:
    $(MAKE) LD="/opt/SUNWhpc/bin/mpf90 -o" \
    LDFLAGS="-stackvar -fsimple=1 -xO3 -xarch=v9b -DHPCF_MPI -lmpi \
             -xlic_lib=sunperf" \
    FC="/opt/SUNWhpc/bin/mpf90 -c" \
    FCFLAGS="-stackvar -fsimple=1 -xO3 -xarch=v9b -xchip=ultra \
             -xlic_lib=sunperf -xalias=actual -fpover -ftrap=%none \
             -fnonstd -libmil -dalign -I/opt/SUNWhpc/HPC5.0/include/v9" \
    EX=$(EX) BINROOT=$(BINROOT) $(TYPE)

#================== HPCx SP4 ==========================================
hpcx:
    $(MAKE) LD="mpxlf90_r -o" LDFLAGS="-O3 -q64 -qmaxmem=-1" \
    FC="mpxlf90_r -qsuffix=f=f90 -c" \
    FCFLAGS="-O3 -q64 -qmaxmem=-1 -qarch=pwr5 -qtune=pwr5 -qnosave" \
    EX=$(EX) BINROOT=$(BINROOT) $(TYPE)

#================== HPCx SP4 - DEBUG ==================================
hpcx-debug:
    $(MAKE) LD="mpxlf90_r -o" LDFLAGS="-g -C -q64 -O0 -lessl -lhmd" \
    FC="mpxlf90_r -qsuffix=f=f90 -c" \
    FCFLAGS="-g -C -q64 -O0 -qarch=pwr5 -qtune=pwr5 -qnosave" \
    EX=$(EX) BINROOT=$(BINROOT) $(TYPE)
```

```
# Default code
#=====================================================================

master: message check $(OBJ_MOD) $(OBJ_ALL)
    $(LD) $(EXE) $(OBJ_MOD) $(OBJ_ALL) $(LDFLAGS)

# Check that a machine has been specified
check:
    @if test "${FC}" = "undefined"; then \
    echo; echo "*** Fortran 90 compiler unspecified!"; \
    echo; echo "*** Please edit your Makefile entries!"; \
    echo; exit 99; \
    fi
    @if test "${LD}" = "undefined"; then \
    echo; echo "*** Fortran 90 Linker-loaDer unspecified!"; \
    echo; echo "*** Please edit your Makefile entries!"; \
    echo; exit 99; \
    fi
    @mkdir -p $(BINROOT)

message:
    @echo "DL_POLY_3 compilation in MPI mode"
    @echo
    @echo "'Use mpi_module' must be commented out in 'comms_module.f90' and 'parallel_fft.f9
    @echo

# Declare rules
#=====================================================================

.f90.o:
    $(FC) $(FCFLAGS) $*.f90

# Declare dependencies
#=====================================================================

$(OBJ_ALL): $(OBJ_MOD)
ewald_spme_forces.o: parallel_fft.o
```

Makefile_SRL1

```
# Master makefile for DL_POLY_3.06 (serial version 1)
#
# Author - I.T.Todorov February 2006
#
#
# Define default settings
#====================================================================

SHELL=/bin/sh

.SUFFIXES:
.SUFFIXES: .f90 .o

BINROOT=../execute
EX=DLPOLY.Y
EXE=$(BINROOT)/$(EX)

TYPE=master

FC=undefined
LD=undefined

# Define object files
#====================================================================

OBJ_MOD = \
    kinds_f90.o mpi_module.o comms_module.o setup_module.o \
    domains_module.o parse_module.o \
    site_module.o config_module.o defects_module.o \
    vdw_module.o metal_module.o tersoff_module.o \
    three_body_module.o four_body_module.o \
    core_shell_module.o constraints_module.o pmf_module.o tethers_module.o \
    bonds_module.o angles_module.o dihedrals_module.o inversions_module.o \
    external_field_module.o statistics_module.o

OBJ_ALL = \
    warning.o error.o \
    numeric_container.o kinetic_container.o spme_container.o \
    scan_field.o scan_config.o scan_control.o set_bound~.o \
    read_control.o \
    vdw_generate.o vdw_table_read.o metal_generate.o \
    tersoff_generate.o dihedrals_14_check.o read_field.o read_config.o \
    tag_legend.o pass_shared_units.o compress_legend.o build_book_intra.o \
    build_excl_intra.o \
    scale_temperature.o update_shared_units.o \
    core_shell_quench.o constraints_tags.o constraints_quench.o \
    pmf_coms.o pmf_tags.o pmf_vcoms.o pmf_quench.o \
```

```
    set_temperature.o \
    vdw_lrc.o metal_lrc.o system_init.o \
    export_atomic_data.o set_halo_particles.o \
    link_cell_pairs.o \
    core_shell_on_top.o \
    deport_atomic_data.o pmf_units_set.o compress_book_intra.o \
    relocate_particles.o \
    metal_ld_collect.o metal_ld_export.o metal_ld_set_halo.o \
    metal_ld_compute.o \
    ewald_spme_forc~s.o \
    metal_forces.o vdw_forces.o ewald_real_forces.o \
    coul_dddp_forces.o coul_cp_forces.o coul_fscp_forces.o \
    coul_rfp_forces.o rdf_collect.o ewald_excl_forces.o \
    ewald_frozen_forces.o two_body_forces.o \
    tersoff_forces.o three_body_forces.o four_body_forces.o \
    core_shell_forces.o tethers_forces.o \
    bonds_forces.o angles_forces.o dihedrals_forces.o inversions_forces.o \
    core_shell_relax.o langevin_forces.o \
    nvt_e_scl.o nvt_b_scl.o \
    \
    external_field_apply_vv.o external_field_correct.o \
    pseudo_vv.o zero_k_optimise_vv.o \
    constraints_shake_vv.o pmf_shake_vv.o \
    constraints_rattle.o pmf_rattle.o \
    nvt_h_scl.o npt_h_scl.o nst_h_scl.o \
    nvt_l1_vv.o nve_1_vv.o \
    nvt_e1_vv.o nvt_b1_vv.o nvt_h1_vv.o \
    npt_b1_vv.o npt_h1_vv.o npt_m1_vv.o \
    nst_b1_vv.o nst_h1_vv.o nst_m1_vv.o \
    \
    external_field_apply_lfv.o \
    pseudo_lfv.o zero_k_optimise_lfv.o \
    constraints_shake_lfv.o pmf_shake_lfv.o \
    nvt_l1_lfv.o nve_1_lfv.o \
    nvt_e1_lfv.o nvt_b1_lfv.o nvt_h1_lfv.o \
    npt_b1_lfv.o npt_h1_lfv.o npt_m1_lfv.o \
    nst_b1_lfv.o nst_h1_lfv.o nst_m1_lfv.o \
    \
    xscale.o core_shell_kinetic.o trajectory_write.o \
    defects_reference_read.o defects_reference_export.o \
    defects_reference_set_halo.o defects_link_cells.o defects_write.o \
    statistics_collect.o z_density_collect.o system_revive.o \
    rdf_compute.o z_density_compute.o statistics_result.o \
    dl_poly.o

# Define MPI-SERIAL files
#=========================================================================

FILES_SERIAL = mpi_module.f90 mpif.h set_bound~.f90 ewald_spme_forc~s.f90
```

```
# Define Velocity Verlet files
#=====================================================================

FILES_VV = \
    external_field_apply_vv.f90 external_field_correct.f90 \
    pseudo_vv.f90 zero_k_optimise_vv.f90 \
    constraints_shake_vv.f90 pmf_shake_vv.f90 \
    constraints_rattle.f90 pmf_rattle.f90 \
    nvt_h_scl.f90 npt_h_scl.f90 nst_h_scl.f90 \
    nve_1_vv.f90  nvt_l1_vv.f90 \
    nvt_e1_vv.f90 nvt_b1_vv.f90 nvt_h1_vv.f90 \
    npt_b1_vv.f90 npt_h1_vv.f90 npt_m1_vv.f90 \
    nst_b1_vv.f90 nst_h1_vv.f90 nst_m1_vv.f90 \
    md_vv.f90

# Define LeapFrog Verlet files
#=====================================================================

FILES_LFV = \
    external_field_apply_lfv.f90 \
    pseudo_lfv.f90 zero_k_optimise_lfv.f90 \
    constraints_shake_lfv.f90 pmf_shake_lfv.f90 \
    nve_1_lfv.f90 nvt_l1_lfv.f90 \
    nvt_e1_lfv.f90 nvt_b1_lfv.f90 nvt_h1_lfv.f90 \
    npt_b1_lfv.f90 npt_h1_lfv.f90 npt_m1_lfv.f90 \
    nst_b1_lfv.f90 nst_h1_lfv.f90 nst_m1_lfv.f90 \
    md_lfv.f90

# Examine targets manually
#=====================================================================

all:
    @echo
    @echo "You MUST specify a target machine!"
    @echo
    @echo "Please examine Makefile for permissible targets!"
    @echo
    @echo "If no target suits your system create your own"
    @echo "using the generic target template provided in"
    @echo "this Makefile at entry 'uknown_machine:'."
    @echo

# Fetch MPI-SERIAL subroutines
#=====================================================================

$(FILES_SERIAL):
    make links_serial
```

```
links_serial:
    @for file in ${FILES_SERIAL} ; do \
    echo linking to $$file ;\
    rm -f $$file ;\
    ln -s SERIAL/$$file $$file ;\
    done

# Fetch the Velocity Verlet subroutines
#=======================================================================

$(FILES_VV):
    make links_vv

links_vv:
    @for file in ${FILES_VV} ; do \
    echo linking to $$file ;\
    rm -f $$file ;\
    ln -s VV/$$file $$file ;\
    done

# Fetch the LeapFrog Verlet subroutines
#=======================================================================

$(FILES_LFV):
    make links_lfv

links_lfv:
    @for file in ${FILES_LFV} ; do \
    echo linking to $$file ;\
    rm -f $$file ;\
    ln -s LFV/$$file $$file ;\
    done

# Clean up the source directory
#=======================================================================

clean:
    rm -f $(OBJ_MOD) $(OBJ_ALL) $(FILES_VV) $(FILES_LFV) $(FILES_SERIAL) *.mod

# Generic target template
#=======================================================================
uknown_machine:
    $(MAKE) LD="path to Fortran 90 Linker-loaDer" \
    LDFLAGS="appropriate flags for LD" \
    FC="path to Fortran 90 compiler" \
    FCFLAGS="appropriate flags for FC" \
    EX=$(EX) BINROOT=$(BINROOT) $(TYPE)

# System specific targets follow:
```

```
#======================================================================

#====================== SUN-ULTRA ====================================
sun-ultra:
    $(MAKE) LD="/opt/SUNWspro/bin/f90 -v -o" \
    LDFLAGS="-O3 -xarch=v8plusa -xchip=ultra" \
    FC="/opt/SUNWspro/bin/f90 -c" \
    FCFLAGS="-O3 -xarch=v8plusa -xchip=ultra" \
    EX=$(EX) BINROOT=$(BINROOT) $(TYPE)

#====================== WINDOWS ======================================
win:
    $(MAKE) LD="f95 -o" \
    LDFLAGS=" " \
    FC="f95 -c" \
    FCFLAGS="-O3" \
    EX=$(EX) BINROOT=$(BINROOT) $(TYPE)


# Default code
#======================================================================

master: message check $(OBJ_MOD) $(OBJ_ALL)
    $(LD) $(EXE) $(OBJ_MOD) $(OBJ_ALL) $(LDFLAGS)

# Check that a machine has been specified
check:
    @if test "${FC}" = "undefined"; then \
    echo; echo "*** Fortran 90 compiler unspecified!"; \
    echo; echo "*** Please edit your Makefile entries!"; \
    echo; exit 99; \
    fi
    @if test "${LD}" = "undefined"; then \
    echo; echo "*** Fortran 90 Linker-loaDer unspecified!"; \
    echo; echo "*** Please edit your Makefile entries!"; \
    echo; exit 99; \
    fi
    @mkdir -p $(BINROOT)

message:
    @echo "DL_POLY_3 compilation in SRL1 mode"
    @echo
    @echo "'Use mpi_module' must be uncommented in 'comms_module.f90'"
    @echo

# Declare rules
#======================================================================

.f90.o:
```

```
    $(FC) $(FCFLAGS) $*.f90

# Declare dependencies
#=====================================================================

$(OBJ_ALL): $(OBJ_MOD)
```

Makefile_SRL2

```
# Master makefile for DL_POLY_3.06 (serial version 2)
#
# Author - I.T.Todorov February 2006
#
#
# Define default settings
#====================================================================

SHELL=/bin/sh

.SUFFIXES:
.SUFFIXES: .f90 .o

BINROOT=../execute
EX=DLPOLY.Y
EXE=$(BINROOT)/$(EX)

TYPE=master

FC=undefined
LD=undefined

# Define object files
#====================================================================

OBJ_MOD = \
    kinds_f90.o mpi_module.o comms_module.o setup_module.o \
    domains_module.o parse_module.o \
    site_module.o config_module.o defects_module.o \
    vdw_module.o metal_module.o tersoff_module.o \
    three_body_module.o four_body_module.o \
    core_shell_module.o constraints_module.o pmf_module.o tethers_module.o \
    bonds_module.o angles_module.o dihedrals_module.o inversions_module.o \
    external_field_module.o statistics_module.o

OBJ_ALL = \
    warning.o error.o \
    numeric_container.o kinetic_container.o spme_container.o \
    scan_field.o scan_config.o scan_control.o set_bounds.o \
    read_control.o \
    vdw_generate.o vdw_table_read.o metal_generate.o \
    tersoff_generate.o dihedrals_14_check.o read_field.o read_config.o \
    tag_legend.o pass_shared_units.o compress_legend.o build_book_intra.o \
    build_excl_intra.o \
    scale_temperature.o update_shared_units.o \
    core_shell_quench.o constraints_tags.o constraints_quench.o \
    pmf_coms.o pmf_tags.o pmf_vcoms.o pmf_quench.o \
```

```
    set_temperature.o \
    vdw_lrc.o metal_lrc.o system_init.o \
    export_atomic_data.o set_halo_particles.o \
    link_cell_pairs.o \
    core_shell_on_top.o \
    deport_atomic_data.o pmf_units_set.o compress_book_intra.o \
    relocate_particles.o \
    metal_ld_collect.o metal_ld_export.o metal_ld_set_halo.o \
    metal_ld_compute.o \
    gpfa_wrap.o parallel_fft.o exchange_grid.o ewald_spme_forces.o \
    metal_forces.o vdw_forces.o ewald_real_forces.o \
    coul_dddp_forces.o coul_cp_forces.o coul_fscp_forces.o \
    coul_rfp_forces.o rdf_collect.o ewald_excl_forces.o \
    ewald_frozen_forces.o two_body_forces.o \
    tersoff_forces.o three_body_forces.o four_body_forces.o \
    core_shell_forces.o tethers_forces.o \
    bonds_forces.o angles_forces.o dihedrals_forces.o inversions_forces.o \
    core_shell_relax.o langevin_forces.o \
    nvt_e_scl.o nvt_b_scl.o \
    \
    external_field_apply_vv.o external_field_correct.o \
    pseudo_vv.o zero_k_optimise_vv.o \
    constraints_shake_vv.o pmf_shake_vv.o \
    constraints_rattle.o pmf_rattle.o \
    nvt_h_scl.o npt_h_scl.o nst_h_scl.o \
    nvt_l1_vv.o nve_1_vv.o \
    nvt_e1_vv.o nvt_b1_vv.o nvt_h1_vv.o \
    npt_b1_vv.o npt_h1_vv.o npt_m1_vv.o \
    nst_b1_vv.o nst_h1_vv.o nst_m1_vv.o \
    \
    external_field_apply_lfv.o \
    pseudo_lfv.o zero_k_optimise_lfv.o \
    constraints_shake_lfv.o pmf_shake_lfv.o \
    nvt_l1_lfv.o nve_1_lfv.o \
    nvt_e1_lfv.o nvt_b1_lfv.o nvt_h1_lfv.o \
    npt_b1_lfv.o npt_h1_lfv.o npt_m1_lfv.o \
    nst_b1_lfv.o nst_h1_lfv.o nst_m1_lfv.o \
    \
    xscale.o core_shell_kinetic.o trajectory_write.o \
    defects_reference_read.o defects_reference_export.o \
    defects_reference_set_halo.o defects_link_cells.o defects_write.o \
    statistics_collect.o z_density_collect.o system_revive.o \
    rdf_compute.o z_density_compute.o statistics_result.o \
    dl_poly.o

# Define MPI-SERIAL files
#=========================================================================

FILES_SERIAL = mpi_module.f90 mpif.h
```

```
# Define Velocity Verlet files
#=====================================================================

FILES_VV = \
    external_field_apply_vv.f90 external_field_correct.f90 \
    pseudo_vv.f90 zero_k_optimise_vv.f90 \
    constraints_shake_vv.f90 pmf_shake_vv.f90 \
    constraints_rattle.f90 pmf_rattle.f90 \
    nvt_h_scl.f90 npt_h_scl.f90 nst_h_scl.f90 \
    nve_1_vv.f90  nvt_l1_vv.f90 \
    nvt_e1_vv.f90 nvt_b1_vv.f90 nvt_h1_vv.f90 \
    npt_b1_vv.f90 npt_h1_vv.f90 npt_m1_vv.f90 \
    nst_b1_vv.f90 nst_h1_vv.f90 nst_m1_vv.f90 \
    md_vv.f90

# Define LeapFrog Verlet files
#=====================================================================

FILES_LFV = \
    external_field_apply_lfv.f90 \
    pseudo_lfv.f90 zero_k_optimise_lfv.f90 \
    constraints_shake_lfv.f90 pmf_shake_lfv.f90 \
    nve_1_lfv.f90 nvt_l1_lfv.f90 \
    nvt_e1_lfv.f90 nvt_b1_lfv.f90 nvt_h1_lfv.f90 \
    npt_b1_lfv.f90 npt_h1_lfv.f90 npt_m1_lfv.f90 \
    nst_b1_lfv.f90 nst_h1_lfv.f90 nst_m1_lfv.f90 \
    md_lfv.f90

# Examine targets manually
#=====================================================================

all:
    @echo
    @echo "You MUST specify a target machine!"
    @echo
    @echo "Please examine Makefile for permissible targets!"
    @echo
    @echo "If no target suits your system create your own"
    @echo "using the generic target template provided in"
    @echo "this Makefile at entry 'uknown_machine:'."
    @echo

# Fetch MPI-SERIAL subroutines
#=====================================================================

$(FILES_SERIAL):
    make links_serial
```

```
links_serial:
    @for file in ${FILES_SERIAL} ; do \
    echo linking to $$file ;\
    rm -f $$file ;\
    ln -s SERIAL/$$file $$file ;\
    done

# Fetch the Velocity Verlet subroutines
#=======================================================================

$(FILES_VV):
    make links_vv

links_vv:
    @for file in ${FILES_VV} ; do \
    echo linking to $$file ;\
    rm -f $$file ;\
    ln -s VV/$$file $$file ;\
    done

# Fetch the LeapFrog Verlet subroutines
#=======================================================================

$(FILES_LFV):
    make links_lfv

links_lfv:
    @for file in ${FILES_LFV} ; do \
    echo linking to $$file ;\
    rm -f $$file ;\
    ln -s LFV/$$file $$file ;\
    done

# Clean up the source directory
#=======================================================================

clean:
    rm -f $(OBJ_MOD) $(OBJ_ALL) $(FILES_VV) $(FILES_LFV) $(FILES_SERIAL) *.mod

# Generic target template
#=======================================================================
uknown_machine:
    $(MAKE) LD="path to Fortran 90 Linker-loaDer" \
    LDFLAGS="appropriate flags for LD" \
    FC="path to Fortran 90 compiler" \
    FCFLAGS="appropriate flags for FC" \
    EX=$(EX) BINROOT=$(BINROOT) $(TYPE)

# System specific targets follow:
```

```
#======================================================================

#====================== SUN-ULTRA ====================================
sun-ultra:
    $(MAKE) LD="/opt/SUNWspro/bin/f90 -v -o" \
    LDFLAGS="-O3 -xarch=v8plusa -xchip=ultra" \
    FC="/opt/SUNWspro/bin/f90 -c" \
    FCFLAGS="-O3 -xarch=v8plusa -xchip=ultra" \
    EX=$(EX) BINROOT=$(BINROOT) $(TYPE)

#====================== WINDOWS ======================================
win:
    $(MAKE) LD="f95 -o" \
    LDFLAGS=" " \
    FC="f95 -c" \
    FCFLAGS="-O3" \
    EX=$(EX) BINROOT=$(BINROOT) $(TYPE)

# Default code
#======================================================================

master: message check $(OBJ_MOD) $(OBJ_ALL)
    $(LD) $(EXE) $(OBJ_MOD) $(OBJ_ALL) $(LDFLAGS)

# Check that a machine has been specified
check:
    @if test "${FC}" = "undefined"; then \
    echo; echo "*** Fortran 90 compiler unspecified!"; \
    echo; echo "*** Please edit your Makefile entries!"; \
    echo; exit 99; \
    fi
    @if test "${LD}" = "undefined"; then \
    echo; echo "*** Fortran 90 Linker-loaDer unspecified!"; \
    echo; echo "*** Please edit your Makefile entries!"; \
    echo; exit 99; \
    fi
    @mkdir -p $(BINROOT)

message:
    @echo "DL_POLY_3 compilation in SRL2 mode"
    @echo
    @echo "'Use mpi_module' must be uncommented in 'comms_module.f90' and 'parallel_fft.f90'
    @echo

# Declare rules
#======================================================================

.f90.o:
    $(FC) $(FCFLAGS) $*.f90
```

```
# Declare dependencies
#=======================================================================

$(OBJ_ALL): $(OBJ_MOD)
ewald_spme_forces.o: parallel_fft.o
```

Appendix D

DL_POLY_3 Error Messages and User Action

Introduction

In this appendix we document the error messages encoded in DL_POLY_3 and the recommended user action. The correct response is described as the **standard user response** in the appropriate sections below, to which the user should refer before acting on the error encountered.

The reader should also be aware that some of the error messages listed below may be either disabled in, or absent from, the public version of DL_POLY_3 . Note that the wording of some of the messages may have changed over time, usually to provide more specific information. The most recent wording appears below.

The Standard User Response

DL_POLY_3 uses FORTRAN 90 dynamic array allocation to set the array sizes at run time. This means that a single executable may be compiled to over all the likely uses of the code. It is not foolproof however. Sometimes an estimate of the required array sizes is difficult to obtain and the calculated value may be too small. For this reason DL_POLY_3 retains array dimension checks and will terminate when an array bound error occurs.

When a dimension error occurs, the **standard user response** is to edit the DL_POLY_3 subroutine SET_BOUNDS. Locate where the variable defining the array dimension is fixed and increase accordingly. To do this you should make use of the dimension information that DL_POLY_3 prints in the OUTPUT file prior to termination. If no information is supplied, simply doubling the size of the variable will usually do the trick. If the variable concerned is defined in one of the support subroutines SCAN_CONFIG, SCAN_FIELD, SCAN_CONTROL you will need to insert a new line in SET_BOUNDS to redefine it - after the relevant subroutine has been called! Finally the code must be recompiled, as in this case it will only be necessary to recompile SET_BOUNDS and not the whole code.

The DL_POLY_3 Error Messages

Message 1: error - word_2_real failure

The semantics in some of the INPUT files is wrong. DL_POLY_3 has tried to read a number but the has found a word in non-number format.

Action:

Look into your INPUT files and correct the semantics where appropriate and resubmit. DL_POLY_3 will have printed out in the OUTPUT file what the found non-uniform word is.

Message 2: error - too many atom types in FIELD (scan_field)

This error arises when DL_POLY_3 scans the FIELD file and discovers that there are too many different types of atoms in the system (i.e. the number of unique atom types exceeds the 1000).

Action:

Increase the number of allowed atom types (mmk) in SCAN_FIELD, recompile and resubmit.

Message 3: error - unknown directive found in CONTROL file

This error most likely arises when a directive is misspelt in the CONTROL file.

Action:

Locate the erroneous directive in the CONTROL file and correct error and resubmit.

Message 4: error - unknown directive found in FIELD file

This error most likely arises when a directive is misspelt or is encountered in an incorrect location in the FIELD file, which can happen if too few or too many data records are included.

Action:

Locate the erroneous directive in the FIELD file and correct error and resubmit.

Message 5: error - unknown energy unit requested

The DL_POLY_3 FIELD file permits a choice of units for input of energy parameters. These may be: electron volts (**ev**); k-calories (**kcal**); k-joules (**kj**); or the DL_POLY_3 internal units (10 J) (**internal**). There is no default value. Failure to specify any of these correctly, or reference to other energy units, will result in this error message. See documentation of the FIELD file.

Action:

Correct energy keyword on **units** directive in FIELD file and resubmit.

Message 6: error - energy unit not specified

A **units** directive is mandatory in the FIELD file. This error indicates that DL_POLY_3 has failed to find the required record.

Action:

Add **units** directive to FIELD file and resubmit.

Message 10: error - too many molecule types specified

This should never happen! This indicates an erroneous FIELD file or corrupted DL_POLY_3 executable. Unlike DL_POLY_2 DL_POLY_3 does not have a set limit on the number of kinds of molecules it can handle in any simulation (this is not the same as the number of molecules).

Action:

Examine FIELD for erroneous directives, correct and resubmit.

Message 11: error - duplicate molecule directive in FIELD file

The number of different types of molecules in a simulation should only be specified once. If DL_POLY_3 encounters more than one **molecules** directive, it will terminate execution.

Action:

Locate the extra **molecule** directive in the FIELD file and remove and resubmit.

Message 12: error - unknown molecule directive in FIELD file

Once DL_POLY_3 encounters the **molecules** directive in the FIELD file, it assumes the following records will supply data describing the intramolecular force field. It does not then expect to encounter directives not related to these data. This error message results if it encounters a unrelated directive. The most probable cause is incomplete specification of the data (e.g. when the **finish** directive has been omitted.)

Action:

Check the molecular data entries in the FIELD file, correct and resubmit.

Message 13: error - molecule species not specified

This error arises when DL_POLY_3 encounters non-bonded force data in the FIELD file, *before* the molecular species have been specified. Under these circumstances it cannot assign the data correctly, and therefore terminates.

Action:

Make sure the molecular data appears before the non-bonded forces data in the FIELD file and resubmit.

Message 14: error - too many unique atom types specified

This should never happen! This error most likely arises when the FIELD file or/and DL_POLY_3 executable are corrupted.

Action:

Recompile the program or recreate the FIELD file. If neither of these works, send the problem to us.

Message 15: error - duplicate vdw potential specified

In processing the FIELD file, DL_POLY_3 keeps a record of the specified short range pair potentials as they are read in. If it detects that a given pair potential has been specified before, no attempt at a resolution of the ambiguity is made and this error message results. See specification of FIELD file.

Action:

Locate the duplication in the FIELD file, rectify and resubmit.

Message 16: error - strange exit from FIELD file processing

This should never happen! It simply means that DL_POLY_3 has ceased processing the FIELD data, but has not reached the end of the file or encountered a **close** directive. P robable cause: corruption of the DL_POLY_3 executable or of the FIELD file. We would be interested to hear of other reasons!

Action:

Recompile the program or recreate the FIELD file. If neither of these works, send the problem to us.

Message 17: error - strange exit from CONTROL file processing

See notes on message 16 above.

Message 18: error - duplicate three-body potential specified

DL_POLY_3 has encountered a repeat specification of a three-body potential in the FIELD file.

Action:

Locate the duplicate entry, remove and resubmit job.

Message 19: error - duplicate four-body potential specified

A 4-body potential has been duplicated in the FIELD file.

Action:

Locate the duplicated four-body potential, remove and resubmit job.

Message 20: error - too many molecule sites specified

This should never happen! This error most likely arises when the FIELD file or/and DL_POLY_3 executable are corrupted.

Action:

Recompile the program or recreate the FIELD file. If neither of these works, send the problem to us.

Message 22: error - unsuitable radial increment in TABLE file

This arises when the tabulated potentials presented in the TABLE file have an increment that is greater than that used to define the other potentials in the simulation. Ideally the increment should be $r_{cut}/(mxgrid-4)$, where r_{cut} is the potential cutoff for the short range potentials and `mxgrid` is the parameter defining the length of the interpolation arrays. An increment less than this is permissible however.

Action:

The tables must be recalculated with an appropriate increment.

Message 23: error - incompatible FIELD and TABLE file potentials

This error arises when the specification of the short range potentials is different in the FIELD and TABLE files. This usually means that the order of specification of the potentials is different. When DL_POLY_3 finds a change in the order of specification, it assumes that the user has forgotten to enter one.

Action:

Check the FIELD and TABLE files. Make sure that you correctly specify the pair potentials in the FIELD file, indicating which ones are to be presented in the TABLE file. Then check the TABLE file to make sure all the tabulated potentials are present in the order the FIELD file indicates.

Message 24: error - end of file encountered in TABLE file

This means the TABLE file is incomplete in some way: either by having too few potentials included, or the number of data points is incorrect.

Action:

Examine the TABLE file contents and regenerate it if it appears to be incomplete. If it look intact, check that the number of data points specified is what DL_POLY_3 is expecting.

Message 25: error - wrong atom type found in CONFIG file

On reading the input file CONFIG, DL_POLY_3 performs a check to ensure that the atoms specified in the configuration provided are compatible with the corresponding FIELD file. This message results if they are not.

Action:

The possibility exists that one or both of the CONFIG or FIELD files has incorrectly specified the atoms in the system. The user must locate the ambiguity, using the data printed in the OUTPUT file as a guide, and make the appropriate alteration.

Message 26: error - neutral group option now redundant

DL_POLY_3 does not have the neutral group option.

Action:

Use the Ewald sum option. (It's better anyway.)

Message 27: error - rigid body option now redundant

DL_POLY_3 does not have a rigid body option.

Action:

Consider using DL_POLY_2 instead.

Message 28: error - PMF option now redundant

DL_POLY_3 does not have the potential of mean force option yet.

Action:

Consider using DL_POLY_2 instead.

Message 30: error - too many chemical bonds specified

This should never happen! This error most likely arises when the FIELD file or/and DL_POLY_3 executable are corrupted.

Action:

Recompile the program or recreate the FIELD file. If neither of these works, send the problem to us.

Message 31: error - too many chemical bonds per domain

DL_POLY_3 sets a limit on the number of chemical bond potentials in the simulated system as a whole. (This number is a function of the total number of chemical bonds in the system and by the number of processing nodes.) Termination results if this number is exceeded.

Action:

Increase `mxbond` in SET_BOUNDS, recompile and resubmit.

Message 32: error - coincidence of particles in core-shell unit

DL_POLY_3 has found a fault in the definition of a core-shell unit in the FIELD file. The same particle has been assigned to the core and shell sites.

Action:

Correct the erroneous entry in FIELD and resubmit.

Message 33: error - coincidence of particles in constraint bond unit

DL_POLY_3 has found a fault in the definition of a constraint bond unit in the FIELD file. The same particle has been assigned to the both sites.

Action:

Correct the erroneous entry in FIELD and resubmit.

Message 34: error - length of constraint bond unit >= real space cutoff (rcut)

DL_POLY_3 has found a constraint bond unit length (FIELD) larger than the real space cutoff (rcut) (CONTROL).

Action:

Increase cutoff in CONTROL or decrease the constraint bondlength in FIELD and resubmit. For small system consider using DL_POLY_2.

Message 35: error - coincidence of particles in chemical bond unit

DL_POLY_3 has found a faulty chemical bond in FIELD (defined between the same particle).

Action:

Correct the erroneous entry in FIELD and resubmit.

Message 38: error - transfer array exceeded in metal_ld_export

This should never happen!

Action:

Consider using `densvar` option in CONTROL for extremely non-equilibrium simulations. Alternatively, increase `mxbuff` in SET_BOUNDS recompile and resubmit. Send the problem to us if this is persistent.

Correct the erroneous entry in FIELD and resubmit.

Message 39: error - density array exceeded in metal_ld_export

This should never happen!

Action:

You might consider using `densvar` option in CONTROL. Send the problem to us if this is persistent.

Message 40: error - too many bond constraints specified

This should never happen!

Action:

Recompile the program or recreate the FIELD file. If neither of these works, send the problem to us.

Message 41: error - too many bond constraints per domain

DL_POLY_3 sets a limit on the number of bond constraints in the simulated system as a whole. (This number is a function of the total number of constraint bonds in the system and by the number of processing nodes.) Termination results if this number is exceeded.

Action:

Increase `mxcons` in SET_BOUNDS, recompile and resubmit.

Message 42: error - undefined direction passed to deport_atomic_data

This should never happen!

Action:

Send the problem to us.

Message 43: error - deport_atomic_data outgoing transfer buffer exceeded

This may happen in extremely non-equilibrium simulations or usually when the potential in use do not hold the system stable.

Action:

Consider using `densvar` option in CONTROL for extremely non-equilibrium simulations. Alternatively, increase `mxbuff` in SET_BOUNDS recompile and resubmit.

Message 44: error - deport_atomic_data incoming transfer buffer exceeded

Action:

See Message 43

Message 45: error - too many atoms in CONFIG file

This should never happen!

Action:

Recompile and resubmit. Send the problem to us if this is persistent.

Message 46: error - undefined direction passed to export_atomic_data

This should never happen!

Action:

Send the problem to us.

Message 47: error - undefined direction passed to metal_ld_export

This should never happen!

Action:

Send the problem to us.

Message 48: error - transfer buffer too small in read_table

Action:

Standard user response. Increase `mxbuff` in SET_BOUNDS recompile and resubmit.

Message 49: error - frozen shell (core-shell) unit specified

The DL_POLY_3 option to freeze the location of an atom (i.e. hold it permanently in one position) is not permitted for the shells in core-shell units.

Action:

Remove the frozen atom option from the FIELD file. Consider using a non-polarisable atom instead.

Message 50: error - too many bond angles specified

This should never happen! This error most likely arises when the FIELD file or/and DL_POLY_3 executable are corrupted.

Action:

Recompile the program or recreate the FIELD file. If neither of these works, send the problem to us.

Message 51: error - too many bond angles per domain

DL_POLY_3 limits the number of valence angle potentials in the system to be simulated (actually, the number to be processed by each node) and checks for the violation of this. Termination will result if the condition is violated.

Action:

Increase the parameter `mxangl` in SET_BOUNDS, recompile and resubmit.

Message 52: error - end of FIELD file encountered

This message results when DL_POLY_3 reaches the end of the FIELD file, without having read all the data it expects. Probable causes: missing data or incorrect specification of integers on the various directives.

Action:

Check FIELD file for missing or incorrect data, correct and resubmit.

Message 53: error - end of CONTROL file encountered

This message results when DL_POLY_3 reaches the end of the CONTROL file, without having read all the data it expects. Probable cause: missing **finish** directive.

Action:

Check CONTROL file, correct and resubmit.

Message 54: error - outgoing transfer buffer exceeded in export_atomic_data

This may happen in extremely non-equilibrium simulations or usually when the potential in use do not hold the system stable.

Action:

Consider using `densvar` option in CONTROL for extremely non-equilibrium simulations. Alternatively, increase `mxbuff` in SET_BOUNDS recompile and resubmit.

Message 55: error - end of CONFIG file encountered

This error arises when DL_POLY_3 attempts to read more data from the CONFIG file than is actually present. The probable cause is an incorrect or absent CONFIG file, but it may be due to the FIELD file being incompatible in some way with the CONFIG file.

Action:

Check contents of CONFIG file. If you are convinced it is correct, check the FIELD file for inconsistencies.

Message 56: error - atomic coordinate array exceeded in export_atomic_data

This may happen in extremely non-equilibrium simulations or usually when the potential in use do not hold the system stable.

Action:

Consider using `densvar` option in CONTROL for extremely non-equilibrium simulations. Alternatively, increase `mxatms` in SET_BOUNDS recompile and resubmit.

Message 57: error - too many core-shell units specified

This should never happen!

Action:

Recompile the program or recreate the FIELD file. If neither of these works, send the problem to us.

Message 58: error - number of atoms in system not conserved

An atom has been lost in transfer between nodes. This should never happen!

Action:

Big trouble. Report problem to authors immediately.

Message 59: error - too many core-shell units per domain

DL_POLY_3 limits the number of core-shell units in the system to be simulated (actually, the number to be processed by each node) and checks for the violation of this. Termination will result if the condition is violated.

Action:

Increase the parameter `mxshl` in SET_BOUNDS, recompile and resubmit.

Message 60: error - too many dihedral angles specified

This should never happen!

Action:

Recompile the program or recreate the FIELD file. If neither of these works, send the problem to us.

Message 61: error - too many dihedral angles per domain

DL_POLY_3 limits the number of dihedral angles in the system to be simulated (actually, the number to be processed by each node) and checks for the violation of this. Termination will result if the condition is violated.

Action:

Increase the parameter `mxdihd` in SET_BOUNDS, recompile and resubmit.

Message 62: error - too many tethered atoms specified

This should never happen!

Action:

Recompile the program or recreate the FIELD file. If neither of these works, send the problem to us.

Message 63: error - too many tethered atoms per domain

DL_POLY_3 limits the number of tethered atoms in the system to be simulated (actually, the number to be processed by each node) and checks for the violation of this. Termination will result if the condition is violated.

Action:

Increase the parameter `mxteth` in SET_BOUNDS, recompile and resubmit.

Message 64: error - incomplete core-shell unit found in build_book_intra

This should never happen!

Action:

Report problem to authors.

Message 65: error - too many excluded pairs specified

This should never happen! This error arises when DL_POLY_3 is identifying the atom pairs that cannot have a pair potential between them, by virtue of being chemically bonded for example (see subroutine BUILD_EXCL_INTRA). Some of the working arrays used in this operation may be exceeded, resulting in termination of the program.

Action:

Contact authors.

Message 66: error - coincidence of particles in bond angle unit

DL_POLY_3 has found a fault in the definition of a bond angle in the FIELD file.

Action:

Correct the erroneous entry in FIELD and resubmit.

Message 67: error - coincidence of particles in dihedral unit

DL_POLY_3 has found a fault in the definition of a dihedral unit in the FIELD file.

Action:

Correct the erroneous entry in FIELD and resubmit.

Message 68: error - coincidence of particles in inversion unit

DL_POLY_3 has found a fault in the definition of a inversion unit in the FIELD file.

Action:

Correct the erroneous entry in FIELD and resubmit.

Message 69: error - too many link cells required in three_body_forces

This should not happen! The calculation of three-body forces in DL_POLY_3 is handled by the link cell algorithm. This error arises if the required number of link cells exceeds the permitted array dimension in the code.

Action:

Consider using `densvar` option in CONTROL for extremely non-equilibrium simulations. Alternatively, increase `mxcell` in SET_BOUNDS recompile and resubmit.

Message 70: error - constraint_quench failure

When a simulation with bond constraints is started, DL_POLY_3 attempts to extract the kinetic energy of the constrained atom-atom bonds arising from the assignment of initial random velocities. If this procedure fails, the program will terminate. The likely cause is a badly generated initial configuration.

Action:

Some help may be gained from increasing the cycle limit, by using the directive **mxshak** in the CONTROL file. You may also consider reducing the tolerance of the SHAKE iteration using the directive **shake** in the CONTROL file. However it is probably better to take a good look at the starting conditions!

Message 71: error - too many metal potentials specified

This should never happen!

Action:

Report to authors.

Message 72: error - too many tersoff potentials specified

This should never happen!

Action:

Report to authors.

Message 72: error - too many inversion potentials specified

This should never happen!

Action:

Report to authors.

Message 74: error - unidentified atom in tersoff potential list

This shows that DL_POLY_3 has encountered and erroneous entry for Tersoff potentials in FIELD.

Action:

Correct FIELD and resubmit.

Message 76: error - duplicate tersoff potential specified

This shows that DL_POLY_3 has encountered and erroneous entry for Tersoff potentials in FIELD.

Action:

Correct FIELD and resubmit.

Message 77: error - too many inversion angles per domain

DL_POLY_3 limits the number of inversion potentials in the system to be simulated (actually, the number to be processed by each node) and checks for the violation of this. Termination will result if the condition is violated.

Action:

Increase the parameter `mxinv` in SET_BOUNDS, recompile and resubmit.

Message 78: error - too many link cells required in tersoff_forces

This should not happen! The calculation of Tersoff forces in DL_POLY_3 is handled by the link cell algorithm. This error arises if the required number of link cells exceeds the permitted array dimension in the code.

Action:

Consider using `densvar` option in CONTROL for extremely non-equilibrium simulations. Alternatively, increase `mxcell` in SET_BOUNDS recompile and resubmit.

Message 79: error - tersoff potential cutoff undefined

This shows that DL_POLY_3 has encountered and erroneous entry for Tersoff potentials in FIELD.

Action:

Correct FIELD and resubmit.

Message 80: error - too many pair potentials specified

This should never happen!

Action:

Report to authors.

Message 81: error - unidentified atom in pair potential list

This shows that DL_POLY_3 has encountered and erroneous entry for vdw or metal potentials in FIELD.

Action:

Correct FIELD and resubmit.

Message 82: error - calculated pair potential index too large

This should never happen! In checking the vdw and metal potentials specified in the FIELD file DL_POLY_3 calculates a unique integer indices that henceforth identify every specific potential within the program. If this index becomes too large, termination of the program results.

Action:

Report to authors.

Message 83: error - too many three-body potentials specified

This should never happen!

Action:

Report to authors.

Message 84: error - unidentified atom in three-body potential list

This shows that DL_POLY_3 has encountered and erroneous entry at tbp definition in FIELD.

Action:

Correct FIELD and resubmit.

Message 85: error - required velocities not in CONFIG file

If the user attempts to start up a DL_POLY_3 simulation with any type of **restart** directive (see description of CONTROL file,) the program will expect the CONFIG file to contain atomic

velocities as well as positions. Termination results if these are not present.

Action:

Either replace the CONFIG file with one containing the velocities, or if not available, remove the **restart ...** directive altogether and let DL_POLY_3 create the velocities for itself.

Message 86: error - calculated three-body potential index too large

This should never happen! DL_POLY_3 has a permitted maximum for the calculated index for any three body potential in the system (i.e. as defined in the FIELD file). If there are m distinct types of atom in the system, the index can possibly range from 1 to $(m^2 * (m - 1))/2$. If the internally calculated index exceeds this number, this error report results.

Action:

Report to authors.

Message 87: error - too many link cells required in four_body_forces

This should not happen! The calculation of four-body forces in DL_POLY_3 is handled by the link cell algorithm. This error arises if the required number of link cells exceeds the permitted array dimension in the code.

Action:

Consider using `densvar` option in CONTROL for extremely non-equilibrium simulations. Alternatively, increase `mxcell` in SET_BOUNDS recompile and resubmit.

Message 88: error - legend array exceeded in build_book_intra

This should never happen! Dimension of legend array exceeded.

Action:

Increase parameter `mxfix` in SET_BOUNDS, recompile and resubmit. If the error persists contact authors.

Message 89: error - too many four-body potentials specified

This should never happen!

Action:

Report to authors.

Message 90: error - fluctuations in the total number of frozen particles

This should never happen!

Action:

Big trouble. Report to authors.

Message 91: error - unidentified atom in four-body potential list

The specification of a four-body potential in the FIELD file has referenced an atom type that is unknown.

Action:

Locate the errant atom type in the four body potential definition in the FIELD file and correct. Make sure this atom type is specified by an `atoms` directive earlier in the file.

Message 95: error - error - rcut > minimum of all half-cell widths

In order for the minimum image convention to work correctly within DL_POLY_3 , it is necessary to ensure that the major cutoff applied to the pair interactions does not exceed half the perpendicular width of the simulation cell. (The perpendicular width is the shortest distance between opposing cell faces.) Termination results if this is detected. In NVE and NVT simulations this can only happen at the start of a simulation, but in NPT and N$\underline{\sigma}$T, it may occur at any time.

Action:

Supply a cutoff that is less than half the cell width. If running constant pressure calculations, use a cutoff that will accommodate the fluctuations in the simulation cell. Study the fluctuations in the OUTPUT file to help you with this.

Message 96: error - incorrect atom totals in metal_ld_set_halo

This should never happen!

Action:

Big trouble. Report to authors.

Message 97: error - cannot have tethered shells

Shells in core-shell units cannot be tethered. The dynamical shell model was not designed to work in conjunction with tethered shells. This error results if both are used in the same simulation.

Action:

Correct FIELD and resubmit.

Message 98: error - incorrect atom assignments metal_ld_set_halo

This should never happen!

Action:

Big trouble. Report to authors.

Message 99: error - cannot have constrained shells

Shells in core-shell units cannot be constrained. The dynamical shell model was not designed to work in conjunction with constraint bonds. This error results if both are used in the same simulation.

Action:

Correct FIELD and resubmit.

Message 100: error - core-shell unit separation > rcut

This could only happen if FILED and CONFIG do not match each other or CONFIG is damaged.

Action:

Regenerate CONFIG (and FIELD) and resubmit.

Message 101: error - calculated four-body potential index too large

This should never happen! DL_POLY_3 has a permitted maximum for the calculated index for any four body potential in the system (i.e. as defined in the FIELD file). If there are m distinct types of atom in the system, the index can possibly range from 1 to $(m^2 * (m + 1) * (m + 2))/6$. If the internally calculated index exceeds this number, this error report results.

Action:

Report to authors.

Message 102: error - rcut < 2*rcter (maximum cutoff for tersoff potentials)

The nature of the Tersoff interaction requires they have at least twice shorter cutoff than the standard pair interctions (or the major system cutoff).

Action:

Decrease Tersoff cutoffs in FIELD or increase cutoff in CONTROL and resubmit.

Message 103: error - parameter mxlshp exceeded in pass_shared_units

Various algorithms (constraint and core-shell ones) require that information about 'shared' atoms be passed between nodes. If there are too many such atoms, the arrays holding the information will be exceeded and DL_POLY_3 will terminate execution.

Action:

Increase the parameter `mxlshp` in SET_BOUNDS, recompile and resubmit.

Message 104: error - arrays listme and lstout exceeded in pass_shared_units

This should not happen! Dimensions of indicated arrays have been exceeded.

Action:

Consider using `densvar` option in CONTROL for extremely non-equilibrium simulations.

Message 105: error - shake algorithm (constraints_shake) failed to converge

The SHAKE algorithm for bond constraints is iterative. If the maximum number of permitted iterations is exceeded, the program terminates. Possible causes include: a bad starting configuration;

too large a time step used; incorrect force field specification; too high a temperature; inconsistent constraints (over-constraint) etc..

Action:

You may try to increase the limit of iteration cycles in the constraint subroutines by using the directive **mxshak** and/or decrease the constraint precision by using the directive **shake** in CONTROL. But the trouble may be much more likely to be cured by careful consideration of the physical system being simulated. For example, is the system stressed in some way? Too far from equilibrium?

Message 106: error - neighbour list array too small in link_cell_pairs

Construction of the Verlet neighbour list in subroutine LINK_CELL_PAIRS non-bonded (pair) force has exceeded the neighbour list array dimensions.

Action:

Consider using `densvar` option in CONTROL for extremely non-equilibrium simulations or increase by hand `mxlist` in SET_BOUNDS.

Message 107: error - too many pairs for rdf look up specified

This should never happen! A possible reason is corruption in FIELD or/and DL_POLY_3 executable.

Action:

Reconstruct FIELD, recompile afresh DL_POLY_3 and resubmit. If the problem persists get in touch with DL_POLY_3 authors.

Message 108: error - unidentified atom in rdf look up list

During reading of rdf look up pairs in FIELD DL_POLY_3 has found an unlisted previously atom type.

Action:

Correct FIELD by either defining the new atom type or changing it to an already defined one in the erroneous line. Resubmit.

Message 109: error - calculated pair rdf index too large

This should never happen! In checking the rdf pairs specified in the FIELD file DL_POLY_3 calculates a unique integer index that henceforth identify every rdf pair within the program. If this index becomes too large, termination of the program results.

Action:

Report to authors.

Message 108: error - duplicate rdf look up pair specified

During reading of rdf look up pairs in FIELD DL_POLY_3 has found a duplicate entry in the list.

Action:

Delete the duplicate line and resubmit.

Message 111: error - incomplete bond constraint found in constraints_tags

This should never happen! DL_POLY_3 has not been able to find an atom in a processor domain or its bordering neighbours.

Action:

Probable cause: link cells too small. Use larger potential cutoff. Contact DL_POLY_3 authors.

Message 113: error - intramolecular bookkeeping arrays exceeded in deport_atomic_data

One or more bookkeeping arrays for site-related interactions have been exceeded.

Action:

You will need to print extra diagnostic data from the DEPORT_ATOMIC_DATA subroutine to find where it has gone wrong.

Message 114: error - legend array exceeded in deport_atomic_data

The array `legend` has been exceeded.

Action:

Try increasing parameter `mxfix` in SET_BOUNDS, recompile and resubmit. Contact DL_POLY_3 authors if the problem persists.

Message 115: error - transfer buffer exceeded in update_shared_units

The transfer buffer has been exceeded.

Action:

Consider increasing parameter `mxbuff` in SET_BOUNDS, recompile and resubmit. Contact DL_POLY_3 authors if the problem persists.

Message 116: error - incorrect atom transfer in update_shared_units

An atom has become misplaced during transfer between nodes.

Action:

This happens when the simulation is very numerically unstable. Consider carefully the physical grounds of your simulation, i.e. are you using the adiabatic shell model for accounting polarisation with too big a timestep or too large control distances for the variable timestep, is the ensemble type NPT or N$\underline{\sigma}$T and the system target temperature too close to the melting temperature?

Message 120: error - invalid determinant in matrix inversion

DL_POLY_3 occasionally needs to calculate matrix inverses (usually the inverse of the matrix of cell vectors, which is of size 3×3). For safety's sake a check on the determinant is made, to prevent inadvertent use of a singular matrix.

Action:

Locate the incorrect matrix and fix it - e.g. are cell vectors correct?

Message 141: error - duplicate metal potential specified

During reading of metal potentials (pairs of atom types) in FIELD DL_POLY_3 has found a duplicate pair of atoms in the list.

Action:

Delete one of the duplicate entries and resubmit.

Message 145: error - no two-body like forces specified

This error arises when there are no two-body like interactions specified in FIELD and CONTROL. I.e. none of the following interactions exists or if does, it has been switched off; any coulombic, vdw, metal, tersoff. In DL_POLY_3 expects that particles will be kept apparat, stay separated and never go through each other due to one of the fore-specified interactions.

Action:

Users must alone take measures to prevent such outcome.

Message 150: error - unknown van der waals potential selected

DL_POLY_3 checks when constructing the interpolation tables for the short ranged potentials that the potential function requested is one which is of a form known to the program. If the requested potential form is unknown, termination of the program results. The most probable cause of this is the incorrect choice of the potential keyword in the FIELD file.

Action:

Read the DL_POLY_3 documentation and find the potential keyword for the potential desired.

Message 150: error - unknown van der waals potential selected

DL_POLY_3 checks when constructing the interpolation tables for the metal potentials that the potential function requested is one which is of a form known to the program. If the requested potential form is unknown, termination of the program results. The most probable cause of this is the incorrect choice of the potential keyword in the FIELD file.

Action:

Read the DL_POLY_3 documentation and find the potential keyword for the potential desired.

Message 170: error - too many variables for statistic array

This error means the statistics arrays appearing in subroutine STATISTICS_COLLECT are too small. This should never happen!

Action:

Contact DL_POLY_3 authors.

Message 200: error - rdf/z-density buffer array too small in system_revive

This error indicates that a global summation buffer array in subroutine SYSTEM_REVIVE is too small, i.e `mxbuff` < `mxgrdf`. This should never happen!

Action:

Contact DL_POLY_3 authors.

Message 300: error - incorrect boundary condition for link-cell algorithms

The use of link cells in DL_POLY_3 implies the use of appropriate boundary conditions. This error results if the user specifies octahedral or dodecahedral boundary conditions, which are only available in DL_POLY_2.

Action:

Correct your boundary condition or consider using DL_POLY_2.

Message 305: error - too few link cells per dimension for many-body and tersoff forces subroutines.

The link cells algorithms for many body and tersoff forces in DL_POLY_3 cannot work with less than 27 link cells. Depending on the cell size and the chosen cut-off, DL_POLY_3 may decide that this minimum cannot be achieved and terminate. This should never happen!

Action:

Decrease many-body and tersoff potentials cutoffs or/and number of nodes or/and increase system size.

Message 307: error - link cell algorithm violation

DL_POLY_3 does not like what you are asking it to do. Probable cause: the cutoff is too large to use link cells in this case.

Action:

Rethink the simulation model; reduce the cutoff or/and number of nodes or/and increase system size.

Message 308: error - link cell algorithm in contention with SPME sum precision

DL_POLY_3 does not like what you are asking it to do. Probable cause: you ask for SPME precision that is not achievable by the current settings of the link cell algorithm.

Action:

Rethink the simulation model; reduce number of nodes or/and SPME sum precission or/and increase cutoff.

Message 340: error - invalid integration option requested

DL_POLY_3 has detected an incompatibility in the simulation instructions, namely that the requested integration algorithm is not compatible with the physical model. It *may* be possible to override this error trap, but it is up to the user to establish if this is sensible.

Action:

This is a non-recoverable error, unless the user chooses to override the restriction.

Message 350: error - too few degrees of freedom

This error can arise if a small system is being simulated and the number of constraints applied is too large.

Action:

Simulate a larger system or reduce the number of constraints.

Message 380: error - simulation temperature not specified or < 1 K

DL_POLY_3 has failed to find a **temp** directive in the CONTROL file.

Action:

Place a **temp** directive in the CONTROL file, with the required temperature specified.

Message 381: error - simulation timestep not specified

DL_POLY_3 has failed to find a **timestep** directive in the CONTROL file.

Action:

Place a **timestep** directive in the CONTROL file, with the required timestep specified.

Message 382: error - simulation cutoff not specified

DL_POLY_3 has failed to find a **cutoff** directive in the CONTROL file.

Action:

Place a **cutoff** directive in the CONTROL file, with the required forces cutoff specified.

Message 387: error - system pressure not specified

The target system pressure has not been specified in the CONTROL file. Applies to NPT simulations only.

Action:

Insert a **press** directive in the CONTROL file specifying the required system pressure.

Message 390: error - npt/nst ensemble requested in non-periodic system

A non-periodic system has no defined volume, hence the NPT algorithm cannot be applied.

Action:

Either simulate the system with a periodic boundary, or use another ensemble.

Message 392: error - too many link cells requested

The number of link cells required for a given simulation exceeds the number allowed for by the DL_POLY_3 arrays. Probable cause: your system has expanded unacceptably much to DL_POLY_3 . This may not be physically sensible!

Action:

Increase the parameter `mxcell`.

Message 402: error - van der waals not specified

The user has not set any cutoff in CONTROL, (`rvdw`) - the van der Waals potentials cutoff is needed in order for DL_POLY_3 to proceed.

Action:

Supply a cutoff value for the van der Waals terms in the CONTROL file using the directive **rvdw**, and resubmit job.

Message 410: error - cell not consistent with image convention

The simulation cell vectors appearing in the CONFIG file are not consistent with the specified image convention.

Action:

Locate the variable `imcon` in the CONFIG file and correct to suit the cell vectors.

Message 414: error - conflicting ensemble options in CONTROL file

DL_POLY_3 has found more than one **ensemble** directive in the CONTROL file.

Action:

Locate extra **ensemble** directives in CONTROL file and remove.

Message 416: error - conflicting force options in CONTROL file

DL_POLY_3 has found incompatible directives in the CONTROL file specifying the electrostatic interactions options.

Action:

Locate the conflicting directives in the CONTROL file and correct.

Message 430: error - integration routine not available

A request for a non-existent ensemble has been made or a request with conflicting options that DL_POLY_3 cannot deal with.

Action:

Examine the CONTROL and FIELD files and remove inappropriate specifications.

Message 432: error - undefined tersoff potential

This shows that DL_POLY_3 has encountered an unfamiliar entry for Tersoff potentials in FIELD.

Action:

Correct FIELD and resubmit.

Message 433: error - rcut must be specified for the Ewald sum precision

When specifying the desired precision for the Ewald sum in the CONTROL file, it is also necessary to specify the real space cutoff `rcut`.

Action:

Place the **cut** directive *before* the **ewald precision** directive in the CONTROL file and rerun.

Message 436: error - unrecognised ensemble

An unknown ensemble option has been specified in the CONTROL file.

Action:

Locate **ensemble** directive in the CONTROL file and amend appropriately.

Message 440: error - undefined angular potential

A form of angular potential has been requested which DL_POLY_3 does not recognise.

Action:

Locate the offending potential in the FIELD file and remove. Replace with one acceptable to DL_POLY_3 if this is possible. Alternatively, you may consider defining the required potential in the code yourself. Amendments to subroutines READ_FIELD and ANGLES_FORCES will be required.

Message 442: error - undefined three-body potential

A form of three body potential has been requested which DL_POLY_3 does not recognise.

Action:

Locate the offending potential in the FIELD file and remove. Replace with one acceptable to DL_POLY_3 if this is reasonable. Alternatively, you may consider defining the required potential in the code yourself. Amendments to subroutines READ_FIELD and THREE_BODY_FORCES will be required.

Message 443: error - undefined four-body potential

DL_POLY_3 has been requested to process a four-body potential it does not recognise.

Action:

Check the FIELD file and make sure the keyword is correctly defined. Make sure that subroutine THREE_BODY_FORCES contains the code necessary to deal with the requested potential. Add the code required if necessary, by amending subroutines READ_FIELD and THREE_BODY_FORCES.

Message 444: error - undefined bond potential

DL_POLY_3 has been requested to process a bond potential it does not recognise.

Action:

Check the FIELD file and make sure the keyword is correctly defined. Make sure that subroutine BONDS_FORCES contains the code necessary to deal with the requested potential. Add the code required if necessary, by amending subroutines READ_FIELD and BONDS_FORCES.

Message 445: error - r_14 > rcut in dihedrals_forces

The 1-4 coulombic scaling for a dihedral angle bonding cannot be performed since the 1-4 distance has exceeded the system short range interaction cutoff, `rcut`, in subroutine DIHEDRAL_FORCES.

Action:

To prevent this error occurring again increase `rcut`.

Message 446: error - undefined electrostatic key in dihedral_forces

The subroutine DIHEDRAL_FORCES has been requested to process a form of electrostatic potential it does not recognise.

Action:

The error arises because the integer key `keyfrc` has an inappropriate value (which should not happen in the standard version of DL_POLY_3). Check that the FIELD file correctly specifies the potential. Make sure the version of DIHEDRAL_FORCES does contain the potential you are specifying. Report the error to the authors if these checks are correct.

Action:

To prevent this error occurring again increase `rvdw`.

Message 448: error - undefined dihedral potential

A form of dihedral potential has been requested which DL_POLY_3 does not recognise.

Action:

Locate the offending potential in the FIELD file and remove. Replace with one acceptable to DL_POLY_3 if this is reasonable. Alternatively, you may consider defining the required potential in the code yourself. Amendments to subroutines READ_FIELD and DIHEDRAL_FORCES (and its variants) will be required.

Message 449: error - undefined inversion potential

A form of inversion potential has been encountered which DL_POLY_3 does not recognise.

Action:

Locate the offending potential in the FIELD file and remove. Replace with one acceptable to DL_POLY_3 if this is reasonable. Alternatively, you may consider defining the required potential in the code yourself. Amendments to subroutines READ_FIELD and INVERSIONS_FORCES will be required.

Message 450: error - undefined tethering potential

A form of tethering potential has been requested which DL_POLY_3 does not recognise.

Action:

Locate the offending potential in the FIELD file and remove. Replace with one acceptable to DL_POLY_3 if this is reasonable. Alternatively, you may consider defining the required potential in the code yourself. Amendments to subroutines READ_FIELD and TETHERS_FORCES will be required.

Message 451: error - three-body potential cutoff undefined

The cutoff radius for a three body potential has not been defined in the FIELD file.

Action:

Locate the offending three-body force potential in the FIELD file and add the required cutoff. Resubmit the job.

Message 452: error - undefined vdw potential

A form of vdw potential has been requested which DL_POLY_3 does not recognise.

Action:

Locate the offending potential in the FIELD file and remove. Replace with one acceptable to DL_POLY_3 if this is reasonable. Alternatively, you may consider defining the required potential in the code yourself. Amendments to subroutines READ_FIELD and VDW_GENERATE will be required.

Message 453: error - four-body potential cutoff undefined

The cutoff radius for a four-body potential has not been defined in the FIELD file.

Action:

Locate the offending four body force potential in the FIELD file and add the required cutoff. Resubmit the job.

Message 454: error - undefined external field

A form of external field potential has been requested which DL_POLY_3 does not recognise.

Action:

Locate the offending potential in the FIELD file and remove. Replace with one acceptable to DL_POLY_3 if this is reasonable. Alternatively, you may consider defining the required potential in the code yourself. Amendments to subroutines READ_FIELD and EXTERNAL_FIELD_APPLY will be required.

Message 452: error - undefined metal potential

A form of metal potential has been requested which DL_POLY_3 does not recognise.

Action:

Locate the offending potential in the FIELD file and remove. Replace with one acceptable to DL_POLY_3 if this is reasonable.

Message 462: error - thermostat friction constant must be $>$ 0

A zero or negative value for the thermostat friction constant has been encountered in the CONTROL file.

Action:

Locate the **ensemble** directive in the CONTROL file and assign a positive value to the time constant.

Message 464: error - thermostat relaxation time constant must be $>$ 0

A zero or negative value for the thermostat relaxation time constant has been encountered in the CONTROL file.

Action:

Locate the **ensemble** directive in the CONTROL file and assign a positive value to the time constant.

Message 466: error - barostat relaxation time constant must be $>$ 0

A zero or negative value for the barostat relaxation time constant has been encountered in the CONTROL file.

Action:

Locate the **ensemble** directive in the CONTROL file and assign a positive value to the time constant.

Message 468: error - r0 too large for snm potential with current cutoff

The specified location (r0) of the potential minimum for a shifted n-m potential exceeds the specified potential cutoff. A potential with the desired minimum cannot be created.

Action:

To obtain a potential with the desired minimum it is necessary to increase the van der Waals cutoff. Locate the **rvdw** directive in the CONTROL file and reset to a magnitude greater than r0. Alternatively adjust the value of r0 in the FIELD file. Check that the FIELD file is correctly formatted.

Message 470: error - n < m in definition of n-m potential

The specification of a n-m potential in the FIELD file implies that the exponent m is larger than exponent n. (Not all versions of DL_POLY_3 are affected by this.)

Action:

Locate the n-m potential in the FIELD file and reverse the order of the exponents. Resubmit the job.

Message 471: error - rcut < 2*rctbp (maximum cutoff for three-body potentials)

The cutoff for the pair interactions is smaller than twice that for the three body interactions. This is a bookkeeping requirement for DL_POLY_3 .

Action:

Either use a smaller three-body cutoff, or a larger pair potential cutoff.

Message 472: error - rcut < 2*rcfbp (maximum cutoff for four-body potentials)

The cutoff for the pair interactions is smaller than twice that for the four body interactions. This is a bookkeeping requirement for DL_POLY_3 .

Action:

Either use a smaller four-body cutoff, or a larger pair potential cutoff.

Message 474: error - shell relaxation cycle limit (100) exceeded

The conjugate gradient method in the relaxed shell model exceeded the iteration limit (100).

Action:

Corrective action is to increase the limit or decrease the the convergence criterion by hand in CORE_SHELL_RELAX and recompile. However, it is unlikely that such measures will cure the problem as it is more likely to lay in the physical description of the system being simulated. For example, are the core-shell spring constants well defined? Is the system being too far from equilibrium?

Message 476: error - shells MUST all HAVE either zero or non-zero masses

The polarisation of ions is accounted via a core-shell model as the shell dynamics is either relaxed - shells have no mass, or adiabatic - all shells have non-zero mass.

Action:

Choose which model you would like to use in the simulated system and adapt the shell masses in FIELD to comply with your choice.

Message 478: error - shake algorithms (constraints & pmf) failed to converge

Your system has both bond and pmf constraints. SHAKE (RATTLE_VV1) is done by combined application of both bond and pmf constraints SHAKE (RATTLE_VV1) in an iterative manner until the pmv constraint virial converges to a constant. No such convergence is achieved.

Action:

See Message 515.

Message 480: error - pmf length > minimum of all half-cell widths

The specified PMF length has exceeded the minimum of all half-cell widths.

Action:

Specify shorter PMF length or increase MD cell dimensions.

Message 484: error - only one potential of mean force permitted

Only one potential of mean force is permitted in FIELD.

Action:

Correct the erroneous entries in FIELD.

Message 486: error - only one of the pmf units is permitted to have frozen atoms

Only one of the PMF units is permitted to have frozen atoms.

Action:

Correct the erroneous entries in FIELD.

Message 488: error - too many pmf constraints per domain

This should not happen.

Action:

Is the use of PMF constraints in your system physically sound?

Message 490: error - local pmf constraint not found locally

This should not happen.

Action:

Is your system physically sound, is your system equilibrated?

Message 492: error - a diameter of a pmf unit > minimum of all half cell widths

The diameter of a PMF unit has exceeded the minimum of all half-cell widths.

Action:

Consider the physical concept you are trying to imply in the simulation. Increase MD cell dimensions.

Message 497: error - pmf_quench failure

Action:

See Message 515

Message 498: error - shake algorithm (pmf_shake) failed to converge

Action:

See Message 515

Message 499: error - rattle algorithm (pmf_rattle) failed to converge

Action:

See Message 515

Message 500: error - pmf unit of zero length is not permitted

PMF unit of zero length is found in FIELD. PMF units are either a single atom or a group of atoms usually forming a chemical molecule.

Action:

Correct the erroneous entries in FIELD.

Message 502: error - pmf unit member found to be present more than once

A PMF unit is a group of unique (distingushed) atoms/sites. No repetion of a site is allowed in a PMF unit.

Action:

Correct the erroneous entries in FIELD.

Message 504: error - cutoff too large for TABLE file

The requested cutoff exceeds the information in the TABLE file.

Action:

Reduce the value of the vdw cutoff (`rvdw`) in the CONTROL file or reconstruct the TABLE file.

Message 506: error - compress_legend failed to sort a legend array

This should never happen!

Action:

Report to authors.

Message 513: error - particle assigned to non-existent domain in read_config

This can only happen if particle coordinates do not match the cell parameters in CONFIG. Probably, due to negligence or numerical inaccuracy inaccuracy in generation of big supercell from a small one.

Action:

Make sure lattice parameters and particle coordinates marry each other. Increase accuracy when generating a supercell.

Message 514: error - allowed image conventions are: 0, 1, 2, 3 and 6

DL_POLY_3 has found unsupported boundary condition specified in CONFIG.

Action:

Correct your boundary condition or consider using DL_POLY_2.

Message 515: error - rattle algorithm (constraints_rattle) failed to converge

The RATTLE algorithm for bond constraints is iterative. If the maximum number of permitted iterations is exceeded, the program terminates. Possible causes include: incorrect force field specification; too high a temperature; inconsistent constraints (over-constraint) etc..

Action:

You may try to increase the limit of iteration cycles in the constraint subroutines by using the directive **mxshak** and/or decrease the constraint precision by using the directive **shake** in CONTROL. But the trouble may be much more likely to be cured by careful consideration of the physical system being simulated. For example, is the system stressed in some way? Too far from equilibrium?

Message 516: error - the number nodes MUST be a power of 2 series number

The SPME development in DL_POLY_3 supports only $2^x \times 2^y \times 2^z$ type of domain decomposition.

Action:

You must ensure DL_POLY_3 execution on number of processors represented as $2^x \times 2^y \times 2^z$.

Message 517: error - allowed configuration levels are: 0, 1 and 2

DL_POLY_3 has found erroneous configuration level in CONFIG.

Action:

Correct the error in CONFIG and rerun.

Message 518: error - control distances for variable timestep not intact

DL_POLY_3 has found the control distances for the variable timestep algorithm to be in contention with each other.

Action:

`mxdis` MUST BE $>$ 2.5$\times$ `mndis`. Correct in CONTROL and rerun.

Message 519: error - REVOLD not readable

If REVOLD exists it is either corrupted or generated by an older version of DL_POLY_3 .

Action:

Change the `restart` option in CONTROL and rerun.

Message 520: error - number of nodes MUST be presentable as a product of three integers

A DL_POLY_3 check during the domain decomposition mapping has been violated.

Action:

You have not provided number of processors represented as $2_x \times 2_y \times 2_z$.

Message 530: error - pseudo thermostat thickness MUST comply with: 2 Angs <= thickness < a quarter of the minimum MD cell width

DL_POLY_3 has found a check violated while reading CONTROL.

Action:

Correct accordingly in CONTROL and resubmit.

Message 540: error - pseudo thermostat MUST only be used in bulk simulations, i.e. imcon MUST be 1, 2 or 3

DL_POLY_3 has found a check violated while reading CONTROL.

Action:

Correct accordingly in CONTROL nve or in CONFIG (`imcon`) and resubmit.

Message 551: error - REFERENCE not found !!!

The `defect` detection option is used in conjunction with `restart` but no REFERENCE file is found.

Action:

Supply a REFERENCE configuartion.

Message 552: error - REFERENCE must contain cell parameters !!!

REFERENCE MUST contain cell parameters i.e. image convention MUST be **imcon** = 1, 2, 3 or 6.

Action:

Supply a properly formatted REFERENCE configuartion.

Message 553: error - REFERENCE is inconsistent !!!

An atom has been lost in transfer between nodes. This should never happen!

Action:

Big trouble. Report problem to authors immediately.

Message 554: error - REFERENCE's format different from CONFIG's !!!

REFERENCE complies to the same rulls as CONFIG with the exception that image convention MUST be **imcon** = 1, 2, 3 or 6.

Action:

Supply a properly formatted REFERENCE configuartion.

Message 555: error - particle assigned to non-existent domain in defects_read_reference

Action:

See Message 513

Message 556: error - too many atoms in REFERENCE file

Action:

See Message 45

Message 557: error - undefined direction passed to defects_reference_export

Action:

See Message 42

Message 558: error - outgoing transfer buffer exceeded in defects_reference_export

Action:

See Message 54

Message 559: error - coordinate array exceeded in defects_reference_export

Action:

See Message 56

Message 560: error - rdef found to be > half the shortest interatomic distance in REFERENCE

The defect detection option relies on a cutoff, rdef, to define the vicinity around a site (defined in REFERENCES) in whitch a particle can claim to occupy the site. Evidently, rdef MUST be < half the shortest interatomic distance in REFERENCE.

Action:

Decrease the value of rdef at directive **defect** in CONTROL.

Message 1000: error - working precision mismatch between FORTRAN 90 and MPI implementation

DL_POLY_3 has failed to match the available modes of MPI precision for real numbers to the defined in sc kinds_f90 FORTRAN 90 working precision `wp` for real numbers. `wp` is a precompile parameter.

Action:

This simply mean that `wp` must have been changed from its original value to something else and the new value is not matched by the `mpi_wp` variable in COMMS_MODULE. It is the user's responsibility to ensure that `wp` and `mpi_wp` are compliant. Make the necessary corrections to sc kinds_f90 and/or COMMS_MODULE.

Message 1001: error - allocation failure in comms_module − > gcheck_vector

DL_POLY_3 has failed to find available memory to allocate an array or arrays, i.e. there is lack of sufficient memory (per node) on the execution machine.

Action:

This may simply mean that your simulation is too large for the machine you are running on. Consider this before wasting time trying a fix. Try using more processing nodes if they are available. If this is not an option investigate the possibility of increasing the heap size for your application. Talk to your systems support people for advice on how to do this.

Message 1002: error - deallocation failure in comms_module − > gcheck_vector

DL_POLY_3 has failed to deallocate an array or arrays, i.e. to free memory that is no longer in use.

Action:

Talk to your systems support people for advice on how to manage this.

Message 1003: error - allocation failure in comms_module − > gisum_vector

Action:

See Message 1001

Message 1004: error - deallocation failure in comms_module − > gisum_vector

Action:

See Message 1002

Message 1005: error - allocation failure in comms_module − > grsum_vector

Action:

See Message 1001

Message 1006: error - deallocation failure in comms_module − > grsum_vector

Action:
See Message 1002

Message 1007: error - allocation failure in comms_module − > gimax_vector

Action:
See Message 1001

Message 1008: error - deallocation failure in comms_module − > gimax_vector

Action:
See Message 1002

Message 1009: error - allocation failure in comms_module − > grmax_vector

Action:
See Message 1001

Message 1010: error - deallocation failure in comms_module − > grmax_vector

Action:
See Message 1002

Message 1011: error - allocation failure in parse_module − > get_record

Action:
See Message 1001

Message 1012: error - deallocation failure in parse_module − > get_record

Action:
See Message 1002

Message 1013: error - allocation failure in angles_module − > allocate_angles_arrays

Action:
See Message 1001

Message 1014: error - allocation failure in bonds_module − > allocate_bonds_arrays

Action:
See Message 1001

Message 1015: error - allocation failure in core_shell_module − > allocate_core_shell_arrays

Action:
See Message 1001

Message 1016: error - allocation failure in statistics_module − > allocate_statitics_arrays

Action:
See Message 1001

Message 1017: error - allocation failure in tethers_module − > allocate_tethers_arrays

Action:
See Message 1001

Message 1018: error - allocation failure in constraints_module − > allocate_constraints_arrays

Action:
See Message 1001

Message 1019: error - allocation failure in external_field_module − > allocate_external_field_arrays

Action:
See Message 1001

Message 1020: error - allocation failure in dihedrals_module − > allocate_dihedrals_arrays

Action:
See Message 1001

Message 1021: error - allocation failure in inversions_module − > allocate_inversion_arrays

Action:
See Message 1001

Message 1022: error - allocation failure in vdw_module − > allocate_vdw_arrays

Action:
See Message 1001

Message 1023: error - allocation failure in metal_module − > allocate_metal_arrays

Action:
See Message 1001

Message 1024: error - allocation failure in three_body_module − > allocate_three_body_arrays

Action:
See Message 1001

Message 1025: error - allocation failure in config_module − > allocate_config_arrays

Action:
See Message 1001

Message 1026: error - allocation failure in site_module − > allocate_site_arrays

Action:
See Message 1001

Message 1027: error - allocation failure in tersoff_module − > alocate_tersoff_arrays

Action:
See Message 1001

Message 1028: error - deallocation failure in angles_module − > deallocate_angles_arrays

Action:
See Message 1002

Message 1029: error - deallocation failure in bonds_module − > deallocate_bonds_arrays

Action:
See Message 1002

Message 1030: error - deallocation failure in core_shell_module − > deallocate_core_shell_arrays

Action:
See Message 1002

Message 1031: error - deallocation failure in tethers_module − > deallocate_tethers_arrays

Action:
See Message 1002

Message 1032: error - deallocation failure in constraints_module − > deallocate_constraints_arrays

Action:
See Message 1002

Message 1033: error - deallocation failure in dihedrals_module − > deallocate_dihedrals_arrays

Action:

See Message 1002

Message 1034: error - deallocation failure in inversions_module − > deallocate_inversions_arrays

Action:

See Message 1002

Message 1035: error - allocation failure in defects_module − > allocate_defects_arrays

Action:

See Message 1001

Message 1036: error - allocation failure in pmf_module − > allocate_pmf_arrays

Action:

See Message 1001

Message 1037: error - deallocation failure in pmf_module − > deallocate_pmf_arrays

Action:

See Message 1002

Appendix E

DL_POLY_3 README

```
DL_POLY_3.06
------------

The code is fully self-contained free format FORTRAN90+MPI.

This version supports ALL features that are in DL_POLY_2.16 with the
exceptions of:
  (1) RIDGID BODIES
  (2) Rhombic Dodecahedral (imcon = 5) and Hexagonal prism
      (imcon = 7) boundary conventions
  (3) Classic Ewald and Hautman Klein Ewald Coulomb evaluations.

No previous DL_POLY_3 feature is deprecated.  All NEW features are
documented in the ‘‘DL_POLY_3 User Manual’’.

Warnings:
---------
  (1) DL_POLY_3 now produces index ordered REVCON and HISTORY files
      which are restartable by DL_POLY_2.  Although such printed
      outputs look unscrambled, the actual printing process is not.
      Unscrambled printing is slightly more expensive than natural
      (scrambled) printing.  The cost time-wise is little, < 1%, but
      HD space-wise is approximately 20% per particle entry. This
      is due to the necessary addition of blanks at the end of data
      record, included to align all textual lines to a constant
      length of 72 characters (+ \n as a new-line character).
  (2) REVIVE files produced by version 2 and 3 are not compatible.
      Restarting runs across different sub versions may not be
      possible.
  (3) Although DL_POLY_3 can be compiled in a serial mode, users are
      strongly advised to consider DL_POLY_2 as a better alternative
      to DL_POLY_3 when simulations are likely to be serial jobs
      or/and for systems containing < 5000 particles.  For such
      systems, with both codes compiled in serial modes, DL_POLY_2
      WILL outperform DL_POLY_3 by a factor of X.  Where X depends
```

```
      strongly on the system size - from 3.1 for a 1000 atom system
      to 1.4 for 5000 atom system, and weakly on the system force-
      field complexity - in favour of DL_POLY_3.

Integration Defaults:
---------------------
The default ensemble is NVE.

The default integration scheme is Trotter derived Velocity Verlet
(VV), although Leapfrog Verlet (LFV) is also available.  VV is
considered superior (to LFV) since:
  (1) integration can be developed in symplectic manner for certain
      ensembles, such as: NVE, NVEk (NVT Evans) as well as all
      Nose-Hoover ensembles (NVT, & NPT & NsT when there is no
      external field applied on the system, otherwise they do not
      conserve the phase space volume) and MTK ensembles (NPT & NsT)
  (2) all ensemble variables are updated synchronously and
      thermodynamic quantities and estimators are exact at the every
      step, whereas in LFV particle velocities and thermostat and
      barostat friction velocities are half an integration time-step
      behind the rest of the ensemble variables and due to this
      certain estimators are approximated at full timestep
  (3) it offers better numerical stability and faster convergence in
      iteration algorithms.

The LFV integration may take less cpu time than the VV one for the
certain ensembles - type of system (constraints) and ensemble
dependent.  Usually LFV is faster than VV when bonds constraints are
present in the system.  The relative performance between the LVF and
VV integration (per timestep) is observed to vary in the limits
*** [LFT(t)-VV(t)]/VV(t) = -7:+15% ***  However, the VV algorithms
treat bond constraints in more precise (symplectic) manner than the
LFV ones and thus produce more accurate constraint dynamics.

Makefiles:
----------
The user may compile the code selecting the appropriate Makefile:

``cp Makefile_MPI Makefile''  (for parallel execution - MPI needed)

or

``cp Makefile_SRLx Makefile'' (for serial execution - no MPI needed)

Note that for serial compilation ``Use mpi_module'' has to be
uncommented in comms_module.f90 (as well as in ``parallel_fft.f90''
if Makefile_SRL2 is used).  Makefile_SRL1 compiles the code with a
memory bound SPME algorithm whereas Makefile_SRL2 with a speed bound
one.
```

```
If there is not an entry for the architecture the user wishes to
compile the code for, the user should advise with the administrator
of the machine.  Needed for compilation are:
  (1) a FORTRAN90 compliant compiler (if the PATH to it is not
      passed to the DEFAULT ENVIRONMENT PATH, then it MUST be
      supplied in Makefile)
  (2) MPI libraries COMPILED for the architecture and the compiler
      to be used (if the PATH to them is not passed to the DEFAULT
      ENVIRONMENT PATH, then it MUST be supplied in Makefile)
  (3) a MAKE command (Makefile interpreter in the system SHELL)

Serial Compilation on Windows:
------------------------------

The best way to get around it is to install cygwin on the system
(http://www.cygwin.com/) to emulate a unix like environment and then
use the make command.  A potential problem, you may encounter, is
that a FORTRAN90 compiler may not pick symbolic links.  To resolve
this, you will have to use hard linking in the Makefile.

Contacts:
---------
Dr. I.T.Todorov at CCLRC Daresbury Laboratory e-mail:
I.T.Todorov@dl.ac.uk

Dr. W.Smith at CCLRC Daresbury Laboratory e-mail:
W.Smith@dl.ac.uk
```

Bibliography

[1] Smith, W., and Forester, T., 1996, *J. Molec. Graphics*, **14**, 136. 2

[2] Todorov, I., and Smith, W., 2004, *Phil. Trans. R. Soc. Lond. A*, **362**, 1835. 2, 115

[3] Smith, W., 1987, *Molecular Graphics*, **5**, 71. 2

[4] Smith, W., 1991, *Comput. Phys. Commun.*, **62**, 229. 2, 4, 115

[5] Smith, W., 1993, *Theoretica. Chim. Acta.*, **84**, 385. 2, 4, 115

[6] Smith, W., and Forester, T. R., 1994, *Comput. Phys. Commun.*, **79**, 52. 2

[7] Smith, W., and Forester, T. R., 1994, *Comput. Phys. Commun.*, **79**, 63. 2, 4, 47

[8] Pinches, M. R. S., Tildesley, D., and Smith, W., 1991, *Molecular Simulation*, **6**, 51. 2, 4, 115

[9] Rapaport, D. C., 1991, *Comput. Phys. Commun.*, **62**, 217. 2, 4, 115

[10] Sutton, A. P., and Chen, J., 1990, *Philos. Mag. Lett.*, **61**, 139. 3, 27, 116, 118

[11] Rafii-Tabar, H., and Sutton, A. P., 1991, *Philos. Mag. Lett.*, **63**, 217. 3, 27, 116, 118

[12] Todd, B. D., and Lynden-Bell, R. M., 1993, *Surf. Science*, **281**, 191. 3, 27, 116, 118

[13] Tersoff, J., 1989, *Phys. Rev. B*, **39**, 5566. 3, 28, 116

[14] van Gunsteren, W. F., and Berendsen, H. J. C. 1987, *Groningen Molecular Simulation (GROMOS) Library Manual*. BIOMOS, Nijenborgh, 9747 Ag Groningen, The Netherlands. Standard GROMOS reference. 3, 12

[15] Mayo, S., Olafson, B., and Goddard, W., 1990, *J. Phys. Chem.*, **94**, 8897. 3, 12, 31, 99

[16] Weiner, S. J., Kollman, P. A., Nguyen, D. T., and Case, D. A., 1986, *J. Comp. Chem.*, **7**, 230. 3, 12

[17] Smith, W., 2003, *Daresbury Laboratory*. 4, 9, 62, 69, 70, 85, 132

[18] Allen, M. P., and Tildesley, D. J. 1989, *Computer Simulation of Liquids*. Oxford: Clarendon Press. 4, 36, 43, 45, 48, 115, 117

[19] Andersen, H. C., 1983, *J. Comput. Phys.*, **52**, 24. 4, 45, 115

[20] Adelman, S. A., and Doll, J., 1976, *J. Chem. Phys.*, **64**, 2375. 4, 44, 48

[21] Evans, D. J., and Morriss, G. P., 1984, *Computer Physics Reports*, **1**, 297. 4, 44, 48

[22] Berendsen, H. J. C., Postma, J. P. M., van Gunsteren, W., DiNola, A., and Haak, J. R., 1984, *J. Chem. Phys.*, **81**, 3684. 4, 44, 48, 54

[23] Hoover, W. G., 1985, *Phys. Rev.*, **A31**, 1695. 4, 44, 48, 52, 54

[24] G.M., M., M.E., T., D.J., T., and M.L., K., 1996, *Molec. Phys.*, **87**, 1117. 4, 44, 54, 60

[25] Vessal, B., 1994, *J. Non-Cryst. Solids*, **177**, 103. 16, 17, 31, 93, 99

[26] Smith, W., Greaves, G. N., and Gillan, M. J., 1995, *J. Chem. Phys.*, **103**, 3091. 16, 17, 31, 93, 99

[27] Allinger, N. L., Y. Y. H. L. J.-H., 1998, *J. Am. Chem. Soc.*, **111**, 8551. 16, 18

[28] Smith, W., 1993, *CCP5 Information Quarterly*, **39**, 14. 17, 20, 23

[29] Ryckaert, J. P., and Bellemans, A., 1975, *Chem. Phys. Lett.*, **30**, 123. 18, 94

[30] Schmidt, M. E., S. S., and A., R. S. Molecular dynamics studies of langmuir monolayers of $f(cf_2)_11cooh$. volume 104, page 2101, 1996. 18, 94

[31] Clarke, J. H. R., Smith, W., and Woodcock, L. V., 1986, *J. Chem. Phys.*, **84**, 2290. 25, 96

[32] Finnis, M. W., and Sinclair, J. E., 1984, *Philos. Mag. A*, **50**, 45. 27

[33] Eastwood, J. W., Hockney, R. W., and Lawrence, D. N., 1980, *Comput. Phys. Commun.*, **19**, 215. 31, 32

[34] Neumann, M., 1985, *J. Chem. Phys.*, **82**, 5663. 35

[35] Fuchs, K., 1935, *Proc. R. Soc., A*, **151**, 585. 37, 39

[36] Essmann, U., Perera, L., Berkowitz, M. L., Darden, T., Lee, H., and Pedersen, L. G., 1995, *J. Chem. Phys.*, **103**, 8577. 38, 118

[37] Fincham, D., and Mitchell, P. J., 1993, *J. Phys. Condens. Matter*, **5**, 1031. 40

[38] Lindan, P. J. D., and Gillan, M. J., 1993, *J. Phys. Condens. Matter*, **5**, 1019. 40

[39] shewchuk, J. R. August 4, 1994, *An Introduction to the Conjugate Gradient Method Without the Agonizing Pain, Edition 1 1/4*. School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213. 40

[40] Ryckaert, J. P., Ciccotti, G., and Berendsen, H. J. C., 1977, *J. Comput. Phys.*, **23**, 327. 45, 115

[41] McCammon, J. A., and Harvey, S. C. 1987, *Dynamics of Proteins and Nucleic Acids*. Cambridge: University Press. 47

[42] Izaguirre, J. A. Langevin stabilisation of multiscale mollified dynamics. In Brandt A., Binder K., B. J., editor, *Multiscale Computational Methods in Chemistry and Physics*, volume 117 of *NATO Science Series: Series III - Computer and System Sciences*, pages 34–47. IOS Press, Amsterdam, 2001. 48, 49

[43] Melchionna, S., Ciccotti, G., and Holian, B. L., 1993, *Molec. Phys.*, **78**, 533. 56

[44] G.M., M., D.J., T., and M.L., K., 1994, *J. Chem. Phys.*, **101**, 4177. 56, 60

[45] Melchionna, S., and Cozzini, S., 1998, *University of Rome.* 69

[46] Hockney, R. W., and Eastwood, J. W. 1981, *Computer Simulation Using Particles.* McGraw-Hill International. 115, 117, 119

[47] Smith, W., 1992, *Comput. Phys. Commun.*, **67**, 392. 115

[48] Smith, W., and Fincham, D., 1993, *Molecular Simulation*, **10**, 67. 115

[49] Bush, I. T., 2000, *Daresbury Laboratory.* 118

Index

algorithm, 4, 43, 75
  RATTLE, 4, 45, 46, 115, 119, 184
  SHAKE, 4, 45, 47, 115, 119, 170
  Verlet, 4, 27, 43–47, 117, 119
  Verlet neighbour list, 117
AMBER, 3, 12, 69, 70
angular restraints, 18

barostat, 4, 77, 180
  Berendsen, 55
  Hoover, 56, 60
boundary conditions, 3, 33, 129
  cubic, 85

CCP5, 2, 9
constraints
  bond, 2, 4, 13, 45–48, 89, 108, 117, 119, 160, 165, 169, 170, 184
  Gaussian, 36, 37, 48
  pmf, 13, 47, 48, 90, 108, 117
  tether, 169
CVS, 5, 6

direct Coulomb sum, 33, 35, 77, 83
distance dependant dielectric, 35, 77, 83
distance restraints, 15
DLPROTEIN, 69
Dreiding, 12

ensemble, 4, 176
  Berendsen NσT, 4, 44, 77, 80
  Berendsen NPT, 4, 44, 80
  Berendsen NVT, 4, 44, 77, 80
  canonical, 48
  Evans NVT, 4, 44, 77, 80
  Hoover NσT, 4, 44, 77, 80
  Hoover NPT, 4, 44, 77, 80
  Hoover NVT, 4, 44, 80
  Langevin NVT, 4, 44, 77, 80
  Martyna-Tuckerman-Klein NσT, 80
  Martyna-Tuckerman-Klein NσT, 77, 80
  Martyna-Tuckerman-Klein NPT, 77, 80
  microcanonical, *see* ensemble,NVE
  MTK NσT, 4, 44
  MTK NPT, 4, 44
  NVE, 4, 44, 48, 77, 80
error messages, 72, 154
Ewald
  optimisation, 70, 71
  SPME, 38, 70, 77, 83
  summation, 36, 37, 66, 71, 79, 82, 115, 118, 177

force field, 3, 12, 13, 20, 70, 117, 118, 156, 171, 184
  AMBER, 3, 12
  DL_POLY, 3, 12
  Dreiding, 3, 12, 31
  GROMOS, 3, 12
force shifted Coulomb sum, 34, 78, 83
FORTRAN 90, 5, 6
FTP, 9, 10

Graphical User Interface, 9, 69, 70, 85
GROMOS, 3, 12

Java GUI, 4, 9

licence, 2
long range corrections
  Sutton-Chen, 28
  van der Waals, 26

parallelisation, 4, 63, 115
  Domain Decomposition, 4
  intramolecular terms, 116, 117
polarisation, 39, 40
  shell model, 3, 12, 33, 39–41, 116, 117
potential
  bond, 3, 69, 92, 108, 116, 119, 159, 178
  bonded, 118, 119
  chemical bond, 3, 12–15, 19, 20, 32, 40, 116, 117
  dihedral, 3, 12, 18–22, 94, 108, 116, 117, 164, 178

electrostatics, 3, 7, 13, 15, 17, 33, 77, 78, 83, 108, 116, 178
external field, 12, 41
four-body, 3, 12, 24, 32, 95, 100, 108, 115, 116, 157, 168, 169, 177, 179
improper dihedral, 3, 12, 20, 21, 116
intermolecular, 64
intramolecular, 24, 32, 64
inversion, 3, 12, 21–23, 32, 95, 100, 116, 117, 166, 178
metal, 3, 13, 24, 27, 95, 116, 118
non-bonded, 3, 13, 69, 70, 78, 87, 92, 93, 95, 116–118, 156
nonbonded, 156
Sutton-Chen, *see* potential,metal, *see* potential,metal, *see* potential,metal
tabulated, 102, 158
Tersoff, 3, 12, 24, 28, 31, 95, 98, 166
tersoff, 116, 118
tether, 3, 12, 23, 24, 108, 116, 117, 164, 179
three-body, 3, 12, 13, 15, 24, 31, 32, 69, 95, 99, 108, 115, 116, 157, 165, 168, 179
valence angle, 3, 12, 13, 15, 16, 22, 31, 32, 69, 91, 93, 108, 116–118, 162
van der Waals, 13, 15, 17, 64, 66, 83, 94, 95, 176

reaction field, 35, 36, 78, 83
rigid body, 45
rigid bond, *see* constraints,bond

shell model, 33
stress tensor, 14, 17, 20, 23, 24, 26, 28, 31, 32, 34–36, 40, 47, 54
sub-directory, 132–135
bench, 7
build, 7
data, 7
execute, 7
java, 7
public, 7
source, 7
utility, 7

thermostat, 4, 41, 77, 180
Nosé-Hoover, 56, 58, 60

units
DL_POLY, 6, 109
energy, 88
pressure, 6, 7, 56, 78, 109
temperature, 78

Verlet neighbour list, 64, 117–119, 171

WWW, 2, 5, 9, 10